



TEAMCENTER

Teamcenter Utilities

Teamcenter 2412

Unpublished work. © 2025 Siemens

This Documentation contains trade secrets or otherwise confidential information owned by Siemens Industry Software Inc. or its affiliates (collectively, "Siemens"), or its licensors. Access to and use of this Documentation is strictly limited as set forth in Customer's applicable agreement(s) with Siemens. This Documentation may not be copied, distributed, or otherwise disclosed by Customer without the express written permission of Siemens, and may not be used in any way not expressly authorized by Siemens.

This Documentation is for information and instruction purposes. Siemens reserves the right to make changes in specifications and other information contained in this Documentation without prior notice, and the reader should, in all cases, consult Siemens to determine whether any changes have been made.

No representation or other affirmation of fact contained in this Documentation shall be deemed to be a warranty or give rise to any liability of Siemens whatsoever.

If you have a signed license agreement with Siemens for the product with which this Documentation will be used, your use of this Documentation is subject to the scope of license and the software protection and security provisions of that agreement. If you do not have such a signed license agreement, your use is subject to the Siemens Universal Customer Agreement, which may be viewed at <https://www.sw.siemens.com/en-US/sw-terms/base/ucal/>, as supplemented by the product specific terms which may be viewed at <https://www.sw.siemens.com/en-US/sw-terms/supplements/>.

SIEMENS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS DOCUMENTATION INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY. SIEMENS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR PUNITIVE DAMAGES, LOST DATA OR PROFITS, EVEN IF SUCH DAMAGES WERE FORESEEABLE, ARISING OUT OF OR RELATED TO THIS DOCUMENTATION OR THE INFORMATION CONTAINED IN IT, EVEN IF SIEMENS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

TRADEMARKS: The trademarks, logos, and service marks (collectively, "Marks") used herein are the property of Siemens or other parties. No one is permitted to use these Marks without the prior written consent of Siemens or the owner of the Marks, as applicable. The use herein of third party Marks is not an attempt to indicate Siemens as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A list of Siemens' Marks may be viewed at: www.plm.automation.siemens.com/global/en/legal/trademarks.html. The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow's idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: support.sw.siemens.com

Send Feedback on Documentation: support.sw.siemens.com/doc_feedback_form

Contents

Getting started with Teamcenter utilities

Before you begin	1-1
Setting up a Teamcenter command-line environment	1-1
Log files produced by Teamcenter	1-3
System log files	1-4
Application log files	1-5
Encrypt a password file for use by daemons	1-5
Manage password files	1-6
Preferences and variables used to control application logging	1-7
Syntax definitions	1-8

Configuration utilities

Preference management	2-1
Data access management	2-21
am_re_diagnostics	2-24
Business Modeler IDE	2-70
bmide_admin_data	2-70
propagation_doctor	2-165
Localization	2-175
Organization	2-186
Teamcenter reporting	2-217
Teamcenter interface	2-228
Column configuration utilities	2-229
Workspace utilities	2-236

Product configuration utilities

Content management	3-1
Product structure maintenance	3-7
cfg0_gen_enforced_cond	3-26
cleanup_orphan_arrangements	3-30
cleanup_psdata	3-36
Effectivity mode	3-144

4th Generation Design and Product Configurator utilities

Workflow utilities

Indexing and search utilities

solrCloudSetUp	6-18

Data sharing utilities

Visualization utilities

harvest_mmv_index	8-1

Manufacturing utilities

upgrade_pv_with_visiblelineinfos utility	9-77
Retrieve a structure's Unique Identifier (UID)	9-80

Service Lifecycle Management utilities

plf_attach	10-1

Classification utilities

About smlutility and sml file format	11-32
Overview of smlutility and SML file format	11-32
Elements of an import/export file	11-33
SML import file and BOM line syntax	11-34
Import/export file example	11-35
smlutility keyword syntax and description	11-39

Query utilities

Maintenance utilities

Installation	13-1
warPackager	13-23
Deployment Center	13-25
generateMiniSoftware	13-25
Audit Manager	13-28
Backup and Recovery	13-49
Dispatcher	13-57
Migration	13-65
Portfolio, Program, and Project Management	13-76
Server manager	13-96
Subscription Manager	13-103
System maintenance	13-107
prune_named_references	13-145
Document management	13-169

Teamcenter Integration for NX utilities

Integration utilities

Linked Data Framework utilities	15-1
Teamcenter Community Collaboration	15-16
Exporting Teamcenter data to Intosite	15-23

Teamcenter Automotive Edition-GM Overlay utilities

Aerospace and Defense utilities

Running CAE utilities

Teamcenter Mechatronics Process Management utilities

Volume and database management utilities

File Management System (FMS) utilities ————— 20-74

Schedule Manager Utilities

schmgt_async_r11 ————— 21-1

Systems Engineering utilities



1. Getting started with Teamcenter utilities

Before you begin

Unless otherwise noted in the description of a utility, the program files associated with the utilities described in this manual are located in the **bin** directory under the Teamcenter application root directory.

Prerequisites	Unless stated otherwise in the description of the utility, you must be a member of the dba group or a group that is granted dba privileges.
Enable the utilities	Most of the command utilities are enabled by user authentication using the user ID and password supplied in the -u and -p arguments. If these arguments are not required, the command is enabled by default. If Security Services single sign-on (SSO) is enabled for your server, the -u and -p arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.
Configure the utilities	For information about configuring the utilities, see Setting up a Teamcenter command-line environment .
Start the utilities	Each utility is initiated by entering its name and optional parameters.

Setting up a Teamcenter command-line environment

All server-side Teamcenter command-line utilities must be run within a Teamcenter environment. A Teamcenter environment has many operating system environment variables configured with information about the license server, database connection, password file locations, and so on.

Setting the Teamcenter environment variables

To create a Teamcenter environment, you must set two environment variables and then run a script to set the rest.

1. Set the value of **TC_ROOT** to the directory where the Teamcenter software is located.

```
set TC_ROOT=C:\apps\tc\tcroot
```

2. Set the value of **TC_DATA** to the directory where the information about the Teamcenter database's is located.

```
set TC_DATA=C:\apps\tc\tcdata
```

- Run the **tc_profilevars** script.

```
call %TC_DATA%\tc_profilevars
```

Your command-line is now ready to run utilities.

Example:

To check the Teamcenter database version in Windows, use the '**ask_version**' argument of the **install** command.

```
set TC_ROOT=C:\apps\tc\tcroot
set TC_DATA=C:\apps\tc\tcdata
call %TC_DATA%\tc_profilevars
install -ask_version
```

Example:

To check the Teamcenter database version in Linux, use the '**ask_version**' argument of the **install** command.

```
TC_ROOT=/apps/tc/tcroot; export TC_ROOT
TC_DATA=/apps/tc/tcdata; export TC_DATA
. $TC_DATA/tc_profilevars
install -ask_version
```

The TC_ROOT directory

The **TC_ROOT** environment variable points to a directory where Teamcenter software is located. Although this software connects to a Teamcenter database, there is no specific database information in this directory.

There can be many **TC_ROOT** directories throughout your site, each one serving a different purpose. Each time you install Teamcenter software, that will be the **TC_ROOT** directory for the software that runs from that directory. For example, if you have 3 different volume servers, the value of **TC_ROOT** for each of those FSC processes will be different.

The TC_DATA directory

The **TC_DATA** environment variable points to a directory which contains the information needed to connect Teamcenter software to a single Teamcenter database. For each Teamcenter database, there must be only one **TC_DATA** directory. This directory is shared and then mounted by each machine with a **TC_ROOT** directory.

Caution:

While it is technically possible to avoid network sharing and make copies of a Teamcenter database's **TC_DATA** directory, it is not a best practice. However, if you do make copies, you *must* track all copies and ensure that they all stay in perfect synchronization with the original. Any discrepancy may lead to data corruption.

The tc_profilevars script

This script sets the environment variables required to connect to a specific Teamcenter database, and it is located in the **TC_DATA** directory.

It sets environment variables which tell the Teamcenter software what kind of RDBMS it is connecting to, where to find it, and its login credentials, as well as where to find the licensing servers, a local copy of the POM schema, the FSC configuration, and so on.

To run it properly, you must set two environment variables first.

TC_ROOT	Represents the Teamcenter software used to connect to a Teamcenter database.
TC_DATA	Represents the Teamcenter database the software connects to.

Provided script on Windows systems

On Windows-based Teamcenter servers, a **Command Prompt** script is available which sets up the environment for you. You will find this in the Windows **Start** menu in the Teamcenter folder.

Log files produced by Teamcenter

Teamcenter utilities often produce log files.

Teamcenter creates two types of log files: system log files and application log files. *System log files* are used to record information about global system events; *application log files* are used to record information about specific Teamcenter applications.

Note:

In this context, *application* means either a Teamcenter application (such as Teamcenter or Structure Manager) or a Teamcenter utility (such as **clearlocks**).

System log files and application log files are stored in different directories and controlled by different preferences. Site preferences control system log files and are write protected. Site preferences must be set in the **tc_preferences.xml** preference file stored in the **TC_DATA** directory. Various user, group, and role preferences control application log files. These preferences can be managed using the **Options**

dialog box, accessed from the **Edit** menu. Use the **Options** dialog box to search for preferences, set preference values, create new preferences, and remove existing preferences.

Best practices

Siemens Digital Industries Software recommends that old log files be deleted. Log files can consume large amounts of hard disk space. Log file size should be monitored periodically and old log files deleted to recover hard disk space. Any required new log files can be recreated after old log files are deleted. Log files can be deleted while users are logged on to Teamcenter.

Siemens Digital Industries Software recommends that directory write permissions are left open. It is extremely important to ensure that directories used to store log file have the required write permissions at all times. For example, it is especially important that Teamcenter be able to write to the **\$TC_LOG** directory if any system logging is enabled. Otherwise, the system repeatedly attempts to write to this directory and performance is affected.

For information about log files, see *Teamcenter Administration*.

System log files

The following log files record information about installation and global system events:

- Installation log file (**installdate-time.log**)

This log file is in the **install** directory under the application root directory. The *date-time* stamp represents the date and time Teamcenter Environment Manager was run. For example, **install0522241627.log** indicates that Teamcenter Environment Manager was run at 4:27 on February 24, 2005.

The **TC_Installation_Logging** environment variable enables installation logging.

- POM utilities log file (**tc_install.log**)

Contains the standard output from the POM utilities called by Teamcenter Environment Manager. This log file is in the **logs** directory under the application root directory. The **logs** directory also contains the logs (and when POM utilities fail, **syslogs**) for utilities that Teamcenter Environment Manager calls.

- System log file (**system.log**)

Contains a record of global system events such as database shutdowns and system startup.

Application log files

Each set of application log files consists of a journal file, monitor file, object log file and a **syslog** file. The **TC_Journaling** preference controls the logging of journal files. **syslog** files are always created and can not be suppressed.

Each application log file name is a concatenation of the application name, the OS process ID (PID), and a descriptive file extension. This ensures application log file names are unique for each session and prevents overwriting valuable troubleshooting information. The following is an example of Structure Manager log file names:

`PSEPID.jnlPSEPID.logPSEPID.syslog`

The following application log files are used by Siemens Digital Industries Software support and development to troubleshoot and debug Teamcenter:

- Journal files (**.jnl**)

Contains diagnostic information and is intended for Siemens Digital Industries Software use only.

- Syslog files (**.syslog**)

Contains diagnostic information and is intended for Siemens Digital Industries Software use only.

- Object log files (**.log**)

Contains a record of Teamcenter objects (users, groups, volumes, and so on) created, modified, or deleted during the application session.

Encrypt a password file for use by daemons

Teamcenter stores passwords in local files using advanced encryption standard (AES) 256-bit encryption. These encrypted passwords are used by Teamcenter utilities, services (daemons) that access the Teamcenter database, and Teamcenter installation tools (Deployment Center and TEM).

During installation, you can specify the path to the directory that contains the password files. The installation tool locks access to the password directory to all users except the operating system user performing the installation.

1. To create an encrypted password file, use the **install** utility with the **-encryptpwf** argument.

You can change the encryption key used to encode the password by creating a **CryptKey** file in the **TC_DATA** directory and providing the key in the file. If this **CryptKey** file exists and contains a valid key (32 bytes or more), this key is used instead of the key from the database.

Note:

Changing the encryption key is not required and not a common practice.

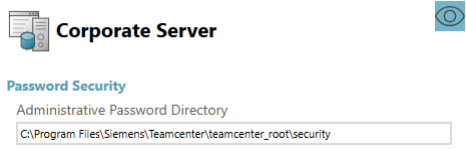
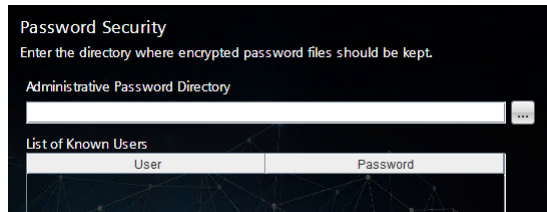
Caution:

The password must not be empty nor contain any whitespace characters such as space, tab, newline, carriage return, form feed, or vertical tab.

In addition, the password must not contain any of the following characters:

! # @ \$ % = & ' " ^ : ; . _ < > () { }

- You can change the directory where password files are stored using your preferred installation tool:

Deployment Center	TEM
In the Corporate Server component properties, click Show all parameters , and then enter the directory in the Administrative Password Directory box. 	Launch TEM in maintenance mode and select Update Security in the Feature Maintenance panel. Proceed to the Password Security panel and enter the directory in the Administrative Password Directory box. 

Changing the stored password does not change the Teamcenter database password. This must be done separately in the Teamcenter client interface.

Caution:

For security reasons, access to the password directory must not be changed unless other methods of ensuring access control are in place.

Manage password files

To provide the best password security, you can store an encrypted password in a designated file and directory location on a local disk. You create the file containing the encrypted password using Teamcenter installation tools (Deployment Center or TEM) or the **install** utility. An environment variable contains the password string to be encrypted. The variable is designated in the installation tool or by an **install** utility argument. The environment variable is not maintained, it is used only during the

encryption process to ensure the clear text password is not persisted. For more information about password encryption, see the **install** utility.

Note:

You can update the encrypted Teamcenter user password using Deployment Center or TEM or the **install** utility. However, this does not change the password in the Teamcenter database. This must be done manually.

The encryption process uses an AES 256-bit encryption key.

For information about managing the encryption key, see *Encrypt a password file for use by daemons*.

The **-pf** argument provides enhanced password security by allowing you to place an encrypted password in a text file and secure the file using operating system-level security. This is stronger security than is provided by the **-p** argument, in which passwords are placed on the utility program command line, allowing a user to run **ps -ef** to display all running utilities and gain access to the utilities' passwords.

The file must contain only the encrypted password. Do not include user names or other text. The password must be one line; new lines and carriage returns are considered a terminator. The password must also be in character encoding consistent with the processes reading it.

You must place the file on a local disk to ensure that access control is managed securely by the operating system representing the file.

- To prepare the password file on Linux, run **chmod 400 file-name**.
- To prepare the password file on Windows, right-click the file and choose **Properties**, and then click the **Security** tab and ensure that **Administrators** is the only group with read access on the local machine.

Note:

File access control becomes complicated when mapping between Linux and Windows platforms.

- On Linux, do not place the password file on disks mounted from other machines. You can run **df** to obtain a list of such disks.
- On Windows, do not place the password file on drives shared with other machines. Using a removal disk (**usb**) ensures that even local administrators only have access when the disk is physically present.

Preferences and variables used to control application logging

The following preferences and environment variables control application logging:

- **TC_Journaling** preference

Globally enables or suppresses creation of all journal files independently of monitor and **syslog** files. This preference can also be set in the rich client interface using the **Edit→Options** command or by using the preference name as an environment variable.

- **CLASSPATH** environment variable

Defines the directory for storing the rich client object log (**.log**) files when the **java.io.tmpdir** key is defined in the Virtual Machine **CLASSPATH** variable, as follows:

```
-Djavaio.tmpdir=path-to-temp-directory
```

Syntax definitions

This manual uses a set of conventions to define the syntax of Teamcenter commands, functions, and values. Following is a sample syntax format:

```
verify_tasks -u=user-id {-p=password | -pf=password-file} [-g=group-name] [-m={list | delete}]  
[-h]
```

The conventions are:

Bold	Bold text represents words and symbols you must enter exactly as shown. For example, you enter verify_tasks exactly as shown.
<i>Italic</i>	Italic text represents values that you supply. For example, you supply your values for <i>user-ID</i> , <i>password</i> and <i>group-name</i> .
	A vertical bar (also called a pipe) represents a choice between mutually exclusive elements. For example, you must specify either the list value or the delete value for the -m argument.
[]	Brackets represent optional elements. For example, the -g= , -m= and -h arguments are optional. (Arguments not in brackets are mandatory. For example, the -u= argument is required.)
{text}	Braces surround mutually exclusive elements that are required. For example, a password value is required. You must use either the -p= or -pf= argument.
...	An ellipsis indicates that you can repeat the preceding element.
text-text	A hyphen separates two words that describe a single value.

For example, *group-name* indicates that you input a single value.

Following are examples of correct syntax for the **verify_tasks** command:

```
$TC_ROOT/bin/verify_tasks -u=dba -p=DBA
$TC_ROOT/bin/verify_tasks -u=dba -p=DBA -m=list
$TC_ROOT/bin/verify_tasks -u=dba -pf=passwords.txt -m=delete
$TC_ROOT/bin/verify_tasks -u=dba -p=DBA -h
```




2. Configuration utilities

Preference management

preferences_manager

This utility can be used to import and export preference information, manage preference definitions and instances, and change preference instance values.

SYNTAX

preferences_manager

```
-u=user-name
{-p=password | -pf=password-file}
-g=group-name
-v=view-contents-in-reports
-mode= {append | category | cleanup | cleanup_definitions | clear | export |
import | migrate | delete | remove | backup | restore }
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled, the username and password are authenticated by SSO rather than Teamcenter. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

append	Appends specified values to an existing preference in the database.
category	Manages preference categories.
cleanup	Cleans up all the stale preferences that were not removed by other administrative tasks (from the UI for instance). Also removes preference values which are inconsistent with preference type.
clear	Clears (removes all) the preferences for the specified scope.
export	Exports scope based preferences in the database to an XML file.
import	Imports specified preference file to the database.
delete	Deletes the specified preference definitions from the system.
remove	Removes the specified preferences for the specified scope from the database.
backup	Creates a backup of preferences in the database for given locations to an output folder.
restore	Restores preferences from specified folder to the database for given locations.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Creation and deletion of preferences is governed by the following rules:

- If you are a user, you can only create or delete user preferences.
- If you are a group administrator, you can create or delete group, role, and user preferences.
- If you are a site administrator, you can create or delete site, group, role, and user preferences.

EXAMPLES

- To import the site preferences in an XML file, skipping the processing for all preferences in the XML file that exist in the database, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba -mode=import
-scope=SITE -file=C:\temp\site_pref.xml -action=SKIP
```

- To import the site preferences in an XML file, overriding the values of preferences in the database with the values assigned to the same preference in the XML file, enter the following command on a single line:

```
preferences_manager -u=user-id -p=password -g=dba -mode=import
-scope=SITE -file=C:\temp\site_pref.xml -action=OVERRIDE
```

- To import the site preferences in an XML file, merging the values of preferences in the database with the values assigned to the same preference in the XML file, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba -mode=import
-scope=SITE -file=C:\temp\site_pref.xml -action=MERGE
```

- To import preferences, overriding the values of the preferences in the database with the values of the same preference in the file, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba -mode=import
-scope=SITE -file=C:\temp\site_pref -action=OVERRIDE
```

- To import a preference (specified on the command line) and override the values in the database with the values specified for the preference on the command line, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba
-mode=import -scope=SITE -file=C:\temp\site_pref
-preference=TestPreference -values=Val1,Val2,Val3 -action=OVERRIDE
```

- To export all user preferences in the database for the user **smith**, enter the following command on a single line:

```
preferences_manager -u=smith -p=password -g=design -mode=export
-scope=USER -out_file=C:\temp\smith.xml
```

Note:

In this example, the utility must be run by the user.

- To export all group preferences in the database for the logged-on group of user **smith**, enter the following command on a single line:

```
preferences_manager -u=smith -p=password -g=design
-mode=export -scope=GROUP -out_file=C:\temp\design.xml
```

- To export preferences specified in an input file, enter the following command on a single line:

```
preferences_manager -u=smith -p=password -g=design
-mode=export -file=C:\temp\input_file.txt
-out_file=c:\temp\exported_preferences.xml
```

- To export user preferences for Teamcenter user **smith**, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba
-mode=export -scope=USER -target=smith -out_file=c:\temp\some-file
```

- To remove the **pref1** and **pref2** preferences for the user **smith**, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba
-mode=remove -scope=USER -target=smith -preferences=pref1,pref2
```

- To remove the **pref1** and **pref2** preferences for the **Engineering** group, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba
-mode=remove -scope=GROUP -target=Engineering
-preferences=pref1,pref2
```

- To append the **HRN_Cavity** value to the existing **Connection:HRN_Core,HRN_GeneralWire** values on the **Connected_ToRules** preference, enter the following command on a single line:

```
preferences_manager -u=Tc-admin-user -p=password -g=dba
-mode=append -scope=SITE -preference=Connected_ToRules
-prefix="Connection:" -values="HRN_Cavity" -delimiter=","
```

-MODE=APPEND

Appends specified values to an existing preference in the database.

```
-mode=append
-scope= {SITE | GROUP | ROLE | USER}
-preference=preference-name
[-prefix=prefix-to-be-searched-in-value]
-values=values-to-be-appended
-delimiter=delimiter-to-be-used-for-append
```

-scope

Specifies the location to which the specified preferences are appended. It is created in the specified location. Valid values are:

- **SITE**

Appends the values to preference at the site location.

- **GROUP**

Appends the values to the preference at the group location.

- **ROLE**

Appends the values to the preference at the role location.

- **USER**

Appends the values to the preference at the user location.

Note:

In all these cases, if the preference does not exist at the location, the preference instance is created at the location with the specified value.

-preference

Specifies the preference to be used for append. If the preference is not found, it is created only if already defined in the database, and only if its protection scope allows for a definition at the given location.

-prefix

Specifies the prefix to be searched in the values. If the prefix is found, the value is appended to it. If the prefix is not specified, values separated by the delimiter are appended individually to the end of the values list.

-values

Specifies the values to append to the existing preference values.

-delimiter

Specifies the delimiter between the values.

[Back to top](#)

-MODE=CATEGORY

Manages preference categories. The list of categories is separated by a delimiter using the **delimiter** option. The default delimiter is a comma.

-mode=category

-action= CREATE

-categories=category-names
 [-**delimiter**=delimiter-used-between-categories]

-mode=category

Manages preference categories.

Note:
 This mode requires system administration privileges.

-action

Specifies the action performed upon the category. The only valid value is **CREATE**. You can delete categories from the **Options** dialog box in the rich client.

-categories

Specifies categories to be created. Categories can be separated by a specified delimiter.

-delimiter

Specifies the delimiter used between categories. If not specified, the default delimiter is a comma.

[Back to top](#)

-MODE=CLEANUP

Removes all instances of all preferences that have no assigned definition. Cleans up user instances of all users with all scopes.

-mode=cleanup

[-**u_target**=user-ID list | -**u_target**=role-ID list | -**u_target**=group-ID list]
 [-**dry_run**]

[-**report_file**=full-path-to-report-file]

-mode=cleanup

Takes actions on preferences as per the given action file.

If the **action_file** parameter is specified, it will proceed with the requested actions on the preference instances at the locations specified through the **u_target**, **r_target** and **g_target** parameters.

The actions are:

- preference value removal
- preference value addition
- preference value replacement
- preference renaming

In case no targets are specified, the described operations are performed on all the site preference instances and on all the users, roles, and groups present in the database.

-u_target

Specifies the list of users for which the action will be performed.

Entries are the ids of users (separated by a comma) for which the logged-in user has privileges.

-r_target

Specifies the list of roles for which the action will be performed.

Entries are the ids of roles (separated by a comma) for which the logged-in user has privileges.

-g_target

Specifies the list of groups for which the action will be performed.

Entries are the ids of groups (separated by a comma) for which the logged-in user has privileges.

-dry_run

If specified, no preferences are removed.

The system only parses the preferences to be removed and prints the findings in the specified report file.

-action_file

Full path to the file that contains all the actions to carry out.

Example:

For a documented example action file, look at
TC_ROOT\install\actions_on_tcxxx_preferences.xml.

-report_file

Full path to the file that will contain logging information on the cleanup task.

If this argument is not provided, the system will create a default file.

[Back to top](#)

-MODE=CLEAR

Removes all preference *instances* in a single scope (user, site, group). If the scope is user, removes only the instances of the target user.

This does not remove any preference *definitions*, not even for custom preferences.

-mode=clear

{[-scope= {SITE | GROUP | ROLE | USER}]


```
[-target=user-ID | role-ID | group-ID]
|
[-u_target=list-of-user-IDs] [-r_target=list-of-role-IDs]
-g_target=list-of-group-IDs]
```

-mode=clear

Removes all preferences of the specified scope.

-scope

Specifies the location from which the specified preferences are cleared. Valid values are:

- **SITE**

Clears only site overlay preferences (that is, only the site preferences that have been modified at a site).

- **GROUP**

Clears all the group preferences of the current logged-on group.

- **ROLE**

Clears all the role preferences of the current logged-on role.

- **USER**

Clears all the user preferences of the current logged-on user.

-target

Specifies the user or role or group ID of the user whose preferences are to be cleared.

This option can only be used with the **-scope** option.

-u_target

Specifies the list of users for which all preference instances are to be cleared. It cannot be used with the **-scope** argument but can be used with the **-r_target** and **-g_target** arguments. Entries are the IDs of users (separated by a comma) for which the logged-on user has privileges.

-r_target

Specifies the list of roles for which all preference instances are to be cleared. It cannot be used with the **-scope** argument but can be used with the **-u_target** and **-g_target** arguments. Entries are the IDs of roles (separated by a comma) for which the logged-on user has privileges.

-g_target

Specifies the list of groups for which all preference instances are to be cleared. It cannot be used with the **-scope** argument, but can be used with the **-r_target** and **-u_target** arguments. Entries are the IDs of groups (separated by a comma) for which the logged-on user has privileges.

[Back to top](#)**-MODE=EXPORT**

Exports all definitions and instances in a specified scope. If the scope is user, only the instances of the target user are included.

```
-mode=export
[-scope= {SITE | GROUP | ROLE | USER}]
[-target= {user-ID | role-ID | group-ID}]
[-file=input-file]
[-categories=comma-separated-categories]
[-delimiter=value-delimiter]
[-out_file=output-file-name]
[-report_file=full-path-to-report-file]
```

-mode=export

Exports scope-based preferences to a specified output file.

Note:

This mode requires system administration privileges.

-scope

Specifies the location from which the specified preferences are exported. Valid values are:

- **SITE**

Only the site preferences matching the specified criteria are considered for export.

- **GROUP**

Only the group preferences (of the current logged-on group) matching the specified criteria are considered for export

- **ROLE**

Only the role preferences (of the current logged-on role) matching the specified criteria are considered for export

- **USER**

Only the user preferences (of the current logged-on user) matching the specified criteria are considered for export.

-target

Specifies the user or role or group ID whose preferences are to be exported. Must be a valid user, role or group ID.

If not specified, the export applies to all preferences (including the site location).

-file

Input file specifying the preferences to be exported. This file contains the preference names (in each line). For example:

Item_show_relations

Item_DefaultChildProperties

-categories

Specifies the categories to export. Categories are a comma-separated list, unless a different delimiter is provided with the **-delimiter** option. This option is ignored if the **-file** option is provided.

-delimiter

Specifies the delimiter to be used for the categories. The default delimiter is a comma if the **-delimiter** option is not specified.

-out_file

Specifies the file to which preferences are exported. The output is generated in **XML** format.

-report_file

Specifies the file that contains logging output. If not specified, a default file is created in the current directory.

[Back to top](#)

-MODE=IMPORT

Imports a specified XML preference file into the database.

```
-mode=import
{[-scope= {SITE | GROUP | ROLE | USER}
[-target= {user-ID | role-ID | group-ID}]
|
[-u_target=list-of-user-IDs] [-r_target=list-of-role-IDs]
[-g_target=list-of-group-IDs]
|
[-file=input-file] [-preview]
[-categories=categories-to-import] [-delimiter=delimiter]
|
[-preference=preference-name] [-values=comma-separated-values]
[-delimiter=delimiter]}
[-protection_scope=default-protection-scope] [-enable_environment=activate]
```

-action= {SKIP | OVERRIDE | MERGE }
-report_file=*file-name*

-mode=*import*

Imports a specified preference file into the database.

-file=*import*

Specifies an XML preference file. All the preferences within the specified file will be imported into the database.

-preview=*import*

Performs a dry run and generates the output onto the console. This option is applicable only if **-file** option is specified.

-scope

Specifies the location under which the preference instances will be imported. This option can possibly be used with the **-target** option, but never with any of the **-u_target**, **-r_target** or **-g_target** options. This option can accept one of the following keywords:

- **SITE**

Preferences in the input file are imported for the site. The logged-in user must be a system administrator.

- **GROUP**

Preferences in the input file are imported for the group of the logged-in user. The logged-in user must at least be a group administrator.

- **ROLE**

Preferences in the input file are imported for the role of the logged-in user. The logged-in user must at least be a role administrator.

- **USER**

Preferences in the input file are imported for the logged-in user.

-target

Specifies the user or role or group ID whose preferences are to be imported. Must be a valid user, role or group ID.

If not specified, the import applies to all preferences (including the site location).

-u_target

Specifies the list of users whose preference instances are to be imported. This argument can be used with the **-r_target** and **-g_target** arguments. Entries are the IDs of users (separated by a comma) for which the logged-on user has privileges.

-r_target

Specifies the list of roles whose preference instances are to be imported. This argument can be used with the **-u_target** and **-g_target** arguments. Entries are the IDs of roles (separated by a comma) for which the logged-on user has privileges.

-g_target

Specifies the list of groups whose preference instances are to be imported. This argument can be used with the **-r_target** and **-u_target** arguments. Entries are the IDs of groups (separated by a comma) for which the logged-on user has privileges.

-preference

Specifies the preference name that has to be imported to the database. This works only when the **-file** option is not specified. The preference must have already been defined in the system for this option to work.

-protection_scope

Specifies the modified (if needed) protection scope to give to the preference specified in the **-preference** option when the preference is a hierarchical preference already defined in the database, and when the logged-in user is a system administrator. Valid values are: Site, Group, Role, and User.

-enable_environment

Activates the specified environment.

-values

Specifies the values for the preference specified in the **-preference** option. This can be comma-separated values if the **-delimiter** option is not specified. In order to specify a delimiter other than comma, use the **-delimiter** option. This option is valid only with the **-preference** option. If needed, values can be surrounded by double quotation marks. For example, "my value".

-delimiter

Specifies the delimiter to be used either for the values (when used in conjunction with the **-values** option) or for the categories (when used with the **-categories** option). The default delimiter is comma if the **-delimiter** option is not specified.

-categories

Specifies the categories to import. Categories are a comma-separated list, unless specified through the **-delimiter** option.

-action

Indicates the action to be taken if a preference exists in the database with a different value. Valid values are:

- **SKIP**

The preference values in the database are untouched.

- **OVERWRITE**

The preference values in the database are overridden with the new values in the input file

- **MERGE**

Merges the values in the database with the values in the input file (that is, the union of values in the database and input file).

-report_file

Specifies the file that to which import results are logged. If not specified, a default file is created. The report file logs the following results:

- That the import file contains only user scope-protected preferences.
- That the import file contains role scope-protected preferences. These preferences were not imported for the given users (and a warning was printed in the output report).
- That the import file contains group scope-protected preferences. These preferences were not imported for the given users/roles (and a warning was printed in the output report).
- That the import file contains some scope-protected preferences or system preferences. These preferences were not imported for the given users/roles/groups (and a warning was printed in the output report).
- That the import file contains preferences that are not yet declared at the site level. These preferences were not imported for the given users/roles/groups (and a warning is printed in the output report).

[Back to top](#)

-MODE=DELETE

Deletes preference definitions. In this mode, the utility deletes single or multiple definitions and their instances.

```
-mode=delete
[-preferences=preference_name-1,preference-name-2, ..., preference-name-n
|
-file=file-name]
[-dry_run]
[-report_file= report-file-name]
```

-mode=delete

Deletes the specified preference definitions from the system.

Note:

This mode requires system administration privileges.

-preferences

Comma separated list of preference names which definitions will be removed from the database.

Names of foundation preferences are not allowed.

This argument cannot be used with the **-file** option.

-file

Specifies a list of preferences to be removed.

Each preference definition to be removed must be on a separate line in the file.

This argument cannot be used with the **-preferences** option.

-dry_run

Does not actually complete the operation, though the information is printed in the report file.

-report_file

Full path to the file that will contain logging information on the upgrade task.

If the argument is not provided, the system will create a default file.

[Back to top](#)

-MODE=REMOVE

Removes instances, not definitions. The exact preference name (or names if removing multiple preferences) must be specified on the command line or in the input file. Only one scope (user, site, group) can be addressed at once. If the scope is user, removes only the specified user preference instances of the target user.

-mode=remove

```
{[-scope= {SITE | GROUP | ROLE | USER}]
[-target= {user-ID | role-ID | group-ID}] |
[-u_target=list-of-user-IDs] [-r_target=list-of-role-IDs]
[-g_target=list-of-group-IDs]}
[-preferences=preference-name]
[-file=file-name]
```

-mode=remove

Removes the specified preference instances from the specified location in the database.

Note:

This mode requires system administration privileges.

-scope

Specifies the location under which the preference instances are deleted. This option may be used with the **-target** option, but never with any of the **-u_target**, **-r_target** or **-g_target** options. This option can accept one of the following keywords:

- **SITE**

Removes the specified preferences if they exist in the current logged-on group preferences list.

- **GROUP**

Removes the specified preferences if they exist in the current logged-on group preferences list.

- **ROLE**

Removes the specified preferences if they exist in the current logged-on role preferences list.

- **USER**

Removes the specified preferences if they exist in the current logged-on user preferences list.

Note:

If not specified, this is the default value.

-target

Specifies the user or role or group ID whose preferences are to be removed. Must be a valid user, role or group ID.

If not specified, the removal applies to all preferences (including the site location).

-preferences

Specifies comma-separated preference names for which the preference instances are to be deleted from the database under the specified locations.

-file

Contains a list of preferences to be deleted. Each preference should be on a separate line in the file.

Note:

This option cannot be used with the **-preferences** option.

[Back to top](#)

-MODE=BACKUP

Creates a backup of preferences in the database for given locations to an output folder.

-mode=backup

Creates backup of preferences in the system. All the SITE preferences will be backed up by default. To create backup of user/role/group instances use **u_list/r_list/g_list** options respectively.

Note:

This mode requires system administration privileges.

-out_folder

Specifies the folder in which preferences backup is created.

The output folder structure is shown below:

- SITE preferences - **<out_folder>\PreferenceBackup_<timestamp>\Site\Site_preferences.xml**
- USER preferences - **<out_folder>\PreferenceBackup_<timestamp>\User\<username>.xml**
- ROLE preferences - **<out_folder>\PreferenceBackup_<timestamp>\Role\<rolename_UID>.xml**
- GROUP preferences - **<out_folder>\PreferenceBackup_<timestamp>\Group\<groupname>.xml**

-u_list

Input file specifying the user IDs whose preference instances are to be backed up. Each user ID must be on a separate line in the file. Must be a valid user ID.

To backup instances of all users provide **u_list** option without any file name.

Example:

1. Create backup for all users in the system

-mode=backup -out_folder=<path-to-output-folder> -u_list

2. Create backup for given users only

-mode=backup -out_folder=<path-to-output-folder> -u_list=<file-having-list-of-user-IDs>

Note:

If **u_list** option is not provided then backup will not be created for any user.

-r_list

Input file specifying the role IDs whose preference instances are to be backed up. Each role ID must be on a separate line in the file. Must be a valid role ID.

To backup instances of all roles provide **r_list** option without any file name.

Example:

1. Create backup for all roles in the system

-mode=backup -out_folder=<path-to-output-folder> -r_list

2. Create backup for given roles only

-mode=backup -out_folder=<path-to-output-folder> -r_list=<file-having-list-of-role-IDs>

Note:

If **r_list** option is not provided then backup will not be created for any role.

-g_list

Input file specifying the group IDs whose preference instances are to be backed up. Each group ID must be on a separate line in the file. Must be a valid group ID.

To backup instances of all groups provide **g_list** option without any file name.

Example:

1. Create backup for all groups in the system

-mode=backup -out_folder=<path-to-output-folder> -g_list

2. Create backup for given groups only

-mode=backup -out_folder=<path-to-output-folder> -g_list=<file-having-list-of-group-IDs>

Note:

If **g_list** option is not provided then backup will not be created for any group.

-report_file

Full path to the file that will contain logging information. If the argument is not provided, the system will create a default file.

[Back to top](#)

-MODE=RESTORE

Restores preferences from specified folder to the database for given locations.

-mode=restore

Restores preference definition and values from the given backup folder. All the Site preferences will be restored to backup state by default. To restore instances of user/role/group use **u_list/r_list/g_list** options respectively. For the preferences that are present in the backup, their definition/values will get restored otherwise if they are not present then those preferences/instances will be removed.

1. If you do not wish to remove the added preferences/values then use **-preference_list** option and provide a file having a list of preferences that you want to restore. Only those preferences will get restored and preferences that are not present in the backup will be preserved.
2. It is recommended to take backup of current system preferences before performing restore operation to recover any data accidentally lost during the restore process.

Note:

This mode requires system administration privileges.

-in_folder

Specifies the folder from which preferences are to be restored. The input folder is created using **-mode=backup**.

-u_list

Input file specifying the user IDs whose preference instances are to be restored. Each user ID must be on a separate line in the file. Must be a valid user ID. To restore all user instances from backup provide **u_list** option without any file name.

Example:

1. Restore all user instances from the backup

-mode=restore -in_folder=<path-to-input-folder> -u_list

2. Restore instances for given users only

-mode=restore -in_folder=<path-to-input-folder> -u_list=<file-having-list-of-user-IDs>

Note:

If **u_list** option is not provided then no user instances will be restored.

-r_list

Input file specifying the role IDs whose preference instances are to be restored. Each role ID must be on a separate line in the file. Must be a valid role ID. To restore all role instances from backup provide **r_list** option without any file name.

Example:

1. Restore all role instances from the backup

-mode=restore -in_folder=<path-to-input-folder> -r_list

2. Restore instances for given roles only

-mode=restore -in_folder=<path-to-input-folder> -r_list=<file-having-list-of-role-IDs>

Note:

If **r_list** option is not provided then no role instances will be restored.

-g_list

Input file specifying the group IDs whose preference instances are to be restored. Each group ID must be on a separate line in the file. Must be a valid group ID. To restore all group instances from backup provide **g_list** option without any file name.

Example:

1. Restore all group instances from the backup

-mode=restore -in_folder=<path-to-input-folder> -g_list

2. Restore instances for given groups only

-mode=restore -in_folder=<path-to-input-folder> -g_list=<file-having-list-of-group-IDs>

Note:

If **g_list** option is not provided then no group instances will be restored.

-preference_list

Input file specifying the list of preferences to be restored from the backup. Each preference name must be on a separate line in the file. Any new preference/instance added in the system that are not present in the backup will be preserved as it is if this option is provided. If this option is not provided then all the preferences from the backup will be restored.

-skip_site

Skips restore operation for site preference values. None of the Site preference values in the database will be changed.

-dry_run

Does not actually complete the operation, though the information is printed in the report file.

-report_file

Full path to the file that will contain logging information. If the argument is not provided, the system will create a default file.

[Back to top](#)

Data access management

am_install_tree

Installs an Access Manager (AM) rule tree at your site. Teamcenter supplies a default rule tree that provides a starting point for creating rules at your site. If you do not specify the **-operation** option, it imports a rule tree. In that case, the **-mode** option is required.

SYNTAX

```
am_install_tree -u=user-id [-p=password | -pf=password-file] [-g=group] [-operation={import | export  
}] -path=file-name [-mode={replace_all | replace_tree | no_replace}] -format={xml | txt} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-operation

Specifies whether to import or export the rule tree file.

If you do not specify the **-operation** option, **am_install_tree** assumes the default of importing a rule tree. In that case, the **-mode** option is required.

=import Default mode. Imports a rule tree.

=export Exports a rule tree.

If you specify **-operation=export**, the **-mode** option is not required.

-path

Specifies the full path of the rule tree file.

-mode

Specifies one of the following installation modes:

=replace_all

Overwrites both the existing rule tree and any named access control lists (ACLs) in the system.

If an ACL exists in the file and does not exist in the rule tree, the ACL gets created.

If the ACL exists in the rule tree and does not exist in the file, then nothing happens to the ACL. This mode does not delete any ACLs.

=replace_tree

Updates the rule tree only. It does not overwrite existing named ACLs.

=no_replace

Default mode. Used only when installing the first rule tree at your site. Does nothing if an existing rule tree is detected.

-format

Specifies one of the following file formats:

=xml

XML file format.

=txt

Text file format.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

RETURN VALUES

Return value 0
upon success

Return value 1
upon failure

EXAMPLES

Following are examples of the use of the **am_install_tree** utility:

- The following example exports the rule tree **export_rule.txt**:

```
am_install_tree -u=Tc_admin -p=password -g=admin
               -operation=export -path=D:\export_rule.txt -format=txt
```

- The following example restores the rule tree to the default:

```
am_install_tree -u=Tc_admin -p=password -g=group
               -path=TC_DATA\tc_am_rule_tree.default
               -mode=replace_all -format=txt
```

am_re_diagnostics

Validates the accuracy of read expressions (REs) stored in the database and compares RE-based verdicts with AM rule tree-based verdicts for all relevant user, group, and role combinations. In case of a discrepancy, this tool reports the associated error.

Syntax

```
am_re_diagnostics [-u=user-ID] {-p=password | -pf=password-file} -g=group]
[-diagnostics_pwd=password]
[-update_pwd=new_password]
[-restore_all_pwd]
[-filePath=filename]
[-h]
```


Arguments

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-diagnostics_pwd

Specifies the password parameter in conjunction with either the list of input users given in the **-filePath** argument, or *all* users if the file path argument is not specified.

Full diagnostics against this list of users is run and all associated groups and roles are used as session input parameters when testing the existing valid REs in the system.

The report is displayed on the console, but it can also be redirected to a file. This file contains the following information in tabular form:

- User, group, role under investigation.
- Total number of relevant REs found.

- Newly found REs to be validated against the rule tree.
- Issues found.
 - If issues are found, the ITK error code is displayed, and further information can be gained from the syslog.
 - If no issues are found, a hyphen (-) is displayed.

Note:

The following important assumptions are made when running this utility:

- Run the **am_read_expression_manager** utility in **onetime** mode as follows to ensure all REs are valid:

```
am_read_expression_manager -u=user -p=pwd -mode=onetime
```

- The RDBMS should be in a *test environment* where user passwords can be changed; the tool expects all users in the list to have the same input password.

Use the command line feature to reset the associated user passwords to a common value (see the **-update_pwd** argument). This is a prerequisite for this diagnostics execution.

- Do not run other process in the RDBMS while this diagnostics test is being executed. Otherwise, the system does not generate or alter existing REs while they are validated by this tool.
- If you execute multiple processes in parallel, partition a large user ID input file into several smaller files with non-overlapping user IDs.
- Do not use this in conjunction with the **-update_pwd** argument, as these two parameters are mutually exclusive.

-update_pwd

Uses the password parameter in conjunction with either the list of input users given in the **-filePath** argument, or *all* users if the file path argument is not specified. This argument updates the password to the password specified in the **cmd**-line option for all such users. As part of this operation, a backup is created with the encrypted values of all passwords about to be changed, so that these original values can be restored once diagnostics execution is complete.

This is required to be able to run the tool in diagnostics mode.

-restore_all_pwd

Restores the state of the database for all users whose password changed as a result of execution of the **-update_pwd** argument described earlier. The backup table created using the **-update_pwd** argument is utilized for this purpose.

-filePath=pathname

Uses an input file containing user IDs of users that needs to be processed. This is applicable to both modes of password change and diagnostics.

-h

Displays help for the utility.

Examples

- To use the **userid** input file to change passwords to **Passw0rd** for all users listed in the input file:

```
am_re_diagnostics -u=user-ID -p=password -g=group
                  -update_pwd=Passw0rd -filePath=path
```

To change passwords to **Passw0rd** for all users:

```
am_re_diagnostics -u=user-ID -p=password -g=group -update_pwd=Passw0rd
```

- To restore original passwords for all users affected by the prior **-update_pwd** command:

```
am_re_diagnostics -u=user-ID -p=password -g=group
                  -restore_all_pwd
```

- To run RE diagnostics for all users in the input file:

```
am_re_diagnostics -u=user-ID -p=password -g=group
                  -diagnostics_pwd=Passw0rd -filePath=path
```

- To run RE diagnostics for all users in the input file:

```
am_re_diagnostics -u=user-ID -p=password -g=group
                  -diagnostics_pwd=Passw0rd -filePath=path
```

am_read_expression_manager

Use this utility to:

- Generate and update Access Manager (AM) read expressions.
- Re-compute out-of-date read expressions.
- Attach read expressions to specified objects.
- Run in daemon mode and keep updating read expressions, as needed, based on changes to the associated objects.

You can execute this utility in one of the following modes:

- **onetime** (standalone)

Processes all dirty read expressions and optionally deletes unreferenced read expressions.

- **daemon**

Processes periodic checks for dirty read expressions, as needed.

- **create**

Provides a file with UIDs resulting from query output or other input parameters in order to generate read expressions for these objects.

- **statistics**

Generates a report with general usage data for read expressions.

Siemens Digital Industries Software recommends best practices to optimize read expression generation using this utility.

SYNTAX

```
am_read_expression_manager [-u=user-ID {-p=password | -pf=password-file} -g=group]
[-mode=onetime | daemon | create | stop | status | statistics]
[-batchSize=size]
[-sleepTime=time-in-seconds]
[-startTime=HH:mm:ss]
[-endTime=HH:mm:ss]
[-filePath=filename]
[-active_project]
[-saved_after=MM-DD-YYYY]
[-dryRun]
[-savedQuery=query]
[-delete_unreferenced]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies the mode for running the utility. Because the report is written to the **syslog** file, set the **TC_KEEP_SYSTEM_LOG** preference to **true** to retain the **syslog** file.

- **onetime**

Exits after one time invocation; there is no repeated processing or **daemon** mode execution.

- **daemon**

Periodically checks for stale read expressions and processes these as needed. After a configurable pause time, the utility is re-executed. (See comments on **batchSize** and **sleepTime**.)

- **create**

Certain objects of interest are associated to read expressions if they do not already exist. Input can be a UID file, saved query or other input parameters. (See documentation for the **-savedQuery**, **-filePath**, **-active_project** and **-saved_after** arguments.)

- **stop**

Stops the read expression daemon gracefully.

- **status**

Returns a status, which validates whether the daemon is running.

- **statistics**

Displays various relevant data related to read expressions, such as counts, number of unique expressions, maximum length, breakdown by class, and so forth.

Statistics output also contains information on counts of unsupported read expressions. For example, the output may include instances of read expressions for objects where unsupported accessor terms are represented. In addition, this output can include parameterized read expressions, where lists of project team UIDs are compacted into a string alias.

-batchSize=size

For **oneTime** and **daemon** modes, this is the maximum size of a batch of objects having their read expressions updated in a single transaction. In these modes, the default size is **10000**.

-sleepTime=time-in-seconds

For the **daemon** mode, this argument specifies the wait time until a new update task is executed. The default is **10** seconds.

-startTime=HH:mm:ss

For the **daemon** mode, this argument specifies the start (00:00:00 to 23:59:59) of the interval during which daemon read expression regeneration is active. A best practice is to use a time interval during which less business operations are executed. The default is unset.

-endTime=HH:mm:ss

For the **daemon** mode, this argument specifies the end (00:00:00 to 23:59:59) of the interval during which daemon read expression regeneration is active. A best practice is to use a time interval during which less business operations are executed. The default is unset.

-filePath=filename

Specifies an input file containing the UIDs of objects that need to be processed. This is relevant to the **create** mode.

-active_project

For the **create** mode, this argument gathers all workspace objects in the active project and applies read expressions if they do not already exist.

-saved_after=MM-DD-YYYY

For the **create** mode, this argument gathers all workspace objects that were the final version saved after the specified date and applies read expressions if they do not already exist.

-dryRun

For the **create** mode, this performs the associated queries, but it only outputs the number of found objects. It does not actually create the read expression associations for them.

Note:

This argument is recommended for any create operation so that the system is protected from potentially very large numbers of found objects for read expressions.

If your query returns more than 1M objects, it is recommended that you either re-formulate or filter the object list further.

-savedQuery=query-name

For the **create** mode, this argument executes the specified saved query and, then applies read expressions to the respective objects using the output list of tags.

Optionally, if the saved query requires input parameters, these can also be specified in sequence using additional matching **-name** and **-value** command line parameters.

-delete_unreferenced

For **daemon** or **oneTime** mode, this argument deletes all unreferenced read expressions, for example, read expressions where the **object_uid** no longer exists in the database.

-h

Displays help for the utility.

READ EXPRESSION MANAGER CONCURRENCY MODE

Read expression processing in concurrency mode (for example, parallel execution) is supported under certain conditions. If the identified number of stale read expressions exceeds the threshold set by the **RE_MGR_concurrency_objs_threshold** preference, the read expression manager process spawns five subprocesses. These sub-processes execute read expression generation in parallel, thereby improving the performance of read expression generation. Once the identified number of stale read expressions drops below the designated threshold, execution reverts back to a single process.

EXAMPLES

- Use a UID input file to generate read expressions:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=create -filePath=path-to-input-file
```

- Create read expressions for all workspace objects in an active project:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=create -active_project
```

- Create read expressions for all workspace objects saved after May 2, 2020:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=create -saved_after=05-02-2020
```

- Create read expressions for all items with a unique ID:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=create -savedQuery="Item ID" -name="Item ID" -value=item_id
```

- Output the number of found objects without creating the read expression associations for them:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=create -saved_after=05-02-2020 -dryRun
```

- Update stale read expressions:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=onetime
```

- Update stale read expressions using a daemon with interval and batch settings, where 7200 seconds equals 2 hours of sleep time between subsequent execution of the daemon update:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=daemon -batchSize=1000 -sleepTime=7200
```

- Update stale read expressions using a daemon with sleep start/end time interval between 2 AM and 4 AM as active time for updates, with a sleep time of 10 seconds:

```
am_read_expression_manager -u=user-ID -p=password -g=group
                           -mode=daemon -batchSize=1000 -sleepTime=10 -startTime=02:00:00
                           -endTime=03:59:59
```

- Output statistics:


```
am_read_expression_manager -u=user-ID -p=password -g=group  
-mode=statistics
```

To exit a running daemon process, you must create a file with content containing the following name and location.

Example:

```
%TC_ROOT%\log\am_read_expr_graceful_exit.tmp
```

The daemon process checks for the existence of this file and exits if found. You must delete this file before you run the daemon process again.

am_rule_test_harness

The **am_rule_test_harness** utility uses an input file that contains test information and outputs the test results embedded inside of the Administration Data Documentation report. While this utility is running, it searches for specified objects and evaluates whether the operations are granted or denied for a given user, group, role, and optionally project combination.

The Administration Data Documentation report also contains the Access Manager rules, Organization, and other administration data types, as each of these types can have an impact on these rules or these rules can impact access to the administration data of these types. To view the outcome of this utility:

1. Open the *output-directory/AdminDataReport/index.html* in a browser.
2. Click **Access Manager**.
3. Click **Test Results**.

The generated output report specifies the conditions that passed and failed. If this report indicates corrections need to be made, correct the data in the input file and rerun the tests using the updated input file.

Note:

When the Access Manager rule tree contains the **Current Group Is** condition, the **am_rule_test_harness** utility uses the group from the current logged-on user and not the group specified in the input XML file.

SYNTAX

```
am_rule_test_harness -u=Tc-admin-user {-p=password | -pf=password-file} [-g=group]
  -inputFile=path-name-to-input-file -outputDir=path-name-to-output-file
  -adminDataTypes -listTypes [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputFile

Specifies the full path to the Access Manager test harness input XML file.

Locate sample input files for Access Manager (**am_rule_test_harness_sample_input.xml**) in **TC_ROOT\bin**.

Note:

To include children of **Form**, you must use the **Form** class for the search criteria.

-outputDir

Specifies the full path to where the Access Manager test harness output file is generated.

The utility also copies all data used for the report into this directory to ensure you have a copy of all related data:

- Report input file
- Report output file
- Exported administrative data from the environment
- Administration Data Documentation report
- Utility log file

-adminDataTypes

Specifies a comma-separated list of administration data types to be exported. Following are the administration data types available for export:

- **AccessManager**
- **Organization**
- **Preferences**
- **Projects**
- **RevisionRules**
- **SavedQueries**
- **Stylesheets**
- **Subscriptions**
- **WorkflowTemplates**

Use **all** as the value to export all administration data types.

-listTypes

Lists all supported administration data types that can be specified in the **-adminDataTypes** option. If specified, this option does not export any administration data or run any test harness test.

-h

Displays help for this utility.

EXAMPLES

The **am_rule_test_harness** utility requires an XML input file, which specifies the user, group, role combination, object, and privileges to be tested. You can also use the **uid** of an object in **searchCriteria** in order to perform rules testing, as shown in the second example input file.

```
am_rule_test_harness -u=johnadmin -p=passjohn -g=dba  
-inputFile=C:\inputDir\am_rule_test_harness_input.xml  
-outputDir=C:\output
```

- Following is a sample input XML file to test Access Manager rules.

```

<TestSuite name="Test Suite 1" description="This test suite will test x,y,z">

  <UserTest description="Tests READ for Manufacturing group on Items is denied" user_id="user2"
    group="Manufacturing" role="Designer" project="" searchCriteria="Item{item_id=id1}">
    <PrivilegeTest privilege="READ" expectedResult="Deny"/>
    <PrivilegeTest privilege="WRITE" expectedResult="Deny"/>
    <PrivilegeTest privilege="DELETE" expectedResult="Grant"/>
    <PrivilegeTest privilege="CICO" expectedResult="Deny"/>
    <PrivilegeTest privilege="EXPORT" expectedResult="Deny"/>
    <PrivilegeTest privilege="IMPORT" expectedResult="Deny"/>
  </UserTest>

  <UserTest description="Tests for Engineering group on Items" user_id="user2"
    group="Engineering" role="Analyst" project="" searchCriteria="Item{item_id=id1}">
    <PrivilegeTest privilege="READ" expectedResult="Deny"/>
    <PrivilegeTest privilege="WRITE" expectedResult="Deny"/>
    <PrivilegeTest privilege="DELETE" expectedResult="Grant"/>
    <PrivilegeTest privilege="CICO" expectedResult="Deny"/>
    <PrivilegeTest privilege="EXPORT" expectedResult="Deny"/>
    <PrivilegeTest privilege="IMPORT" expectedResult="Deny"/>
  </UserTest>
</TestSuite>

```

- An input file can use the **uid** of an object in the **searchCriteria** argument for rules testing. The input format for **uid** in **searchCriteria** is:

searchCriteria="uid=uid1,uid=uid2"

The following input file checks the read and write privilege of the object whose **uid** is **Btd9zn1s58c_TA**.

```

<UserTest description="Test read write access for user" user_id="tcadmin" group="tcadmin"
role="dba" project="" searchCriteria="uid=Btd99zn1s58c_TA">
<PrivilegeTest privilege="READ" expectedResult="Grant"/>|
<PrivilegeTest privilege="WRITE" expectedResult="Deny"/>
</UserTest>

```

Following is a list of access privileges:

ADD CONTENT	ADMINISTER ADA LICENSES	ASSIGN TO PROJECT
BATCH PRINT	CHANGE	CHANGE OWNER
CICO (also known as check-in/ check-out)	COPY	CREATE
DELETE	DEMOTE	DIGITALLY SIGN
EFFECTIVITY	EXPORT	IMPORT

IP ADMIN	IP CLASSIFIER	ITAR ADMIN
ITAR CLASSIFIER	MANAGE VARIABILITY	PDF CONTROL
PROMOTE	PUBLISH	READ
REMOTE CHECKOUT	REMOVE CONTENT	REMOVE FROM PROJECT
SUBSCRIBE	TRANSFER IN	TRANSFER OUT
TRANSLATION	UNMANAGE	VOID DIGITAL SIGNATURE
VIEW MARKUP	WRITE	WRITE CLASSIFICATION ICO

ada_util

Provides an alternate procedure to the user interface-based setting of security classification/clearance and license information so it can be called from scripts. This utility, which supports batch mode processing using an **input file**, supports the following authorized data access (ADA) modes:

- **newlicense**
- **modlicense**
- **addlicense**
- **removelicense**
- **adduser**
- **removeuser**
- **addgroup**
- **removegroup**
- **setclassification**
- **setclearance**

To perform debugging, you must:

```
set env variable DEBUG=ON
set DEBUG_FILE_NAME=debug-file
```

For more information about ADA licenses, see *Security Administration*.

PUID

The **ada_util** utility supports using an object's puid instead of the item_id

SYNTAX

```
ada_util [-u=user-id {-p=password | -pf=password-file} g=group]
[-m=mode]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges or another user with **IP Admin** or **ITAR Admin** privilege according to the classification/clearance or license type involved in the operation.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-m

Specifies the mode, which can be one or more of the following:

- **newlicense**
- **modlicense**
- **addlicense**
- **removelicense**
- **adduser**
- **removeuser**
- **addgroup**
- **removegroup**
- **setclassification**
- **setclearance**
- **newlicense** mode

Creates a license with given ID using the **-license_id** and **-type** argument with the optional **-date** expiration date, the optional **-reason** reason, the optional **-lock_date** lock date, and the optional **-qualifying_cfr** string. The **-qualifying_cfr** argument is applicable only to ITAR licenses.

To use the **newlicense** mode, enter the following:

newlicense

Command line options:

```

-license_id=license-id
[-date=expiration-date]
[-reason=reason]
[-type=[itar|ip|exclude]
[-lock_date=lock-date]
[-qualifying_cfr=string]
[-category=string]
[-citizenships=citizenship-list]
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v

```

File options:

```

[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v

```

- **modlicense** mode

Modifies a license identified using the **-license_id** argument to have a new expiration date specified using the **-date** argument, a new lock date specified using the **-lock_date** argument, and new qualifying Code of Federal Regulation (CFR) information specified for licenses of ITAR type using the **-qualifying_cfr** argument.

To use the **modlicense** mode, enter the following:

modlicense**Command line options:**

```

-license_id=license-id
[-date=expiration-date]
[-lock_date=lock-date]
[-qualifying_cfr=string]
[-category=string]
[-citizenships=citizenship-list]
[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v

```

File options:

```
[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v
```

- **addlicense** mode

Adds the license identified using the **-license_id** argument to the object specified by **-Item_id**, **-RevID**, **-Object-Type**, and **-Object-Name** arguments.

Additionally, qualifying paragraph information can be supplied using the **-ead_paragraph** argument.

To use the **addlicense** mode, enter the following:

addlicense**Command line options:**

```
-license_id=license-id
[-ead_paragraph=string]
-Item_id=item [-RevID=revision]
[-Object-Type=object-type -Object-Name=object-name]]
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v
```

File options:

```
[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v
```

- **removelicense** mode

Removes (detaches) a license.

To use the **removelicense** mode, enter the following:

removelicense**Command line options:**

```

-license_id=license-id
[-cfg]=configuration_file]
[-o]=output_file]
[-report_file]=report_file.report]
-v

```

File options:

```

[-filename]=input_data_file.txt]
[-cfg]=configuration_file]
[-o]=output_file]
[-r]=repeat_file.rep]
[-report_file]=report_file.report]
-v

```

- **adduser** mode

Adds the user identified using the **-userid** argument to the license identified by the **-license_id** argument.

To use the **adduser** mode, enter the following:

adduser**Command line options:**

```

-license_id=license-id
-userid=user-id
[-cfg]=configuration_file]
[-o]=output_file]
[-report_file]=report_file.report]
-v

```

File options:

```

[-filename]=input_data_file.txt]
[-cfg]=configuration_file]
[-o]=output_file]
[-r]=repeat_file.rep]
[-report_file]=report_file.report]
-v

```

- **removeuser** mode

Removes (detaches) the user.

To use the **removeuser** mode, enter the following:

removeuser

Command line options:

```
-license_id=license-id  
-userid=user-id  
[-cfg=configuration_file]  
[-o=output_file]  
[-report_file=report_file.report]  
-v
```

File options:

```
[-filename=input_data_file.txt]  
[-cfg=configuration_file]  
[-o=output_file]  
[-r=repeat_file.rep]  
[-report_file=report_file.report]  
-v
```

- **addgroup** mode

Adds the group identified using the **-groupid** argument to the license identified by the **-license_id** argument.

To use the **addgroup** mode, enter the following:

addgroup

Command line options:

```
-license_id=license-id  
-groupid=group-id  
[-cfg=configuration_file]  
[-o=output_file]  
[-report_file=report_file.report]  
-v
```

File options:

```

[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v

```

- **removegroup** mode

Removes (detaches) the group.

To use the **removegroup** mode, enter the following:

removegroup

Command line options:

```

-license_id=license-id
-groupid=group-id
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v

```

File options:

```

[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v

```

- **setclassification** mode

Sets a classification for a given workspace object as specified by the **-classification**, **-Item_id**, **-RevID**, **-Object-Type**, **-Object-Name**, and **-type** arguments.

To use the **setclassification** mode, enter the following:

setclassification

Command line options:

```

-classification=classification
[-ead_paragraph=string]
-Item_id=item [-RevID=revision]

```

```
[-Object-Type=object-type -Object-Name=object-name]]
-type=[itar|ip]
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v
```

File options:

```
[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v
```

- **setclearance** mode

Sets a clearance level for a given user specified by the **-clearance**, **-userid**, and **-type** arguments.

To use the **setclearance** mode, enter the following:

setclearance

Command line options:

```
-clearanceclearance-level
-userid=user-id
-type=[itar|ip]
[-cfg=configuration_file]
[-o=output_file]
[-report_file=report_file.report]
-v
```

File options:

```
[-filename=input_data_file.txt]
[-cfg=configuration_file]
[-o=output_file]
[-r=repeat_file.rep]
[-report_file=report_file.report]
-v
```

-category

Specifies the category type (a string of 128 bytes) for an ADA license. This argument can only be used with the **newlicense** or **modlicense** modes.

-cfg

Specifies the *configuration_file* that is the file path and name of the configuration file.

-citizenships

Specifies the user citizenships for an ADA license. Each citizenship is a two-letter country code from ISO 3166. Multiple citizenships are separated by a comma delimiter, such as:

```
-citizenships=US,GB
```

-classification

Specifies a classification according to context.

-clearance

Specifies a clearance according to context.

-date

Specifies a date for use in **newlicense** and **modlicense** modes.

The default date format is defined as *numericday-abbreviatedmonth-numericyear hours:minutes*. *hours* are in 24-hour format. For example, *11-dec-2006 15:20*. The month names and default date format may change with locale.

-ead_paragraph

Specifies the authorizing paragraph information (a string of 80 characters) recorded on the workspace object specified by **-Item_id**, **-RevID**, **-Object-Type**, and **-Object-Name** arguments, while attaching an **ITAR_License** object identified by the **-license_id** argument.

-filename

Specifies the *input_file* that is the file path and name of the input file.

-h

Displays help for this utility.

-Item_id

Specifies the item to use in finding an object. See the description for the **-Object-Name** argument.

-license_id

Specifies a unique license ID for use with the **newlicense**, **addlicense**, and **modlicense** modes.

-lock_date

Enables authorized users to freeze or unfreeze the license specified by the **-license_id** argument on the specified date. This argument can only be used with the **newlicense** or **modlicense** modes.

-o

Specifies the *output_file* that is the file path and name of the output log file. If not supplied, the default filename is *input_data_file.log* and will be located in the same directory as the *input_data_file*.

-Object-Name

Specifies an object to which the various ADA operations apply. The value should be the object name, for example, a dataset name. This argument is required.

This argument also requires the **-Item_id**, **-RevID**, and **-Object-Type** arguments to provide a basis for finding the object; first, uniquely identifying the item (**-Item_id**), second, identifying the specific revision of that item (**-RevID**), and third, filtering objects attached to the item revision based on object type and name (**-Object-Type**).

- If you specify only the **-Item_id** argument, the object is the item.
- If you specify the **-Item_id** and **-RevID** arguments, the object is the revision.
- If you specify the **-Item_id**, **-RevID**, and **-Object-Type** arguments, the object is that named with the type on the item revision.

See **setclassification** and **addlicense** modes.

-Object-Type

Specifies the object type to filter specification attachments of the given item/revision when finding an object. See the description for the **-Object-Name** argument.

-qualifying_cfr

Specifies the information for the **In Accordance With** attribute (a string of 80 characters) for the **ITAR_License** object identified by the **-license_id** argument. This argument can only be used in with the **newlicense** or **modlicense** modes.

-r

Specifies the repeat file with the extension **.rep** that contains a copy of any data lines that caused an error. This file is overwritten on subsequent run.

-reason

Specifies a string when using the **newlicense** mode.

-report_file

Specifies the report file name with the extension **.report**. This option turns off all create and update options effectively just writing a report of objects found and not found in the specified report file.

-RevID

Specifies the revision within a given item to use in finding an object. See the description for the **-Object-Name** argument.

-type

Specifies the license type as **itar** (International Traffic in Arms Regulations), **ip** (intellectual property), or **exclude** (exclude license) to exclude certain users who are not allowed to see the data.

If **-type** is set to **itar**, the **setclassification** mode applies to **gov_classification** and the **setclearance** mode applies to **gov_clearance**.

If **-type** is set to **ip**, the **setclassification** mode applies to **ip_classification** and the **setclearance** mode applies to **ip_clearance**.

The default value is **ip**.

-userid

Specifies a user for the **setclearance** and **adduser** modes. This argument should not be confused with the login user argument, **-u**.

-v

Turns on verbose mode to get detailed output about the additional propagation relations being processed. If **-report_file** is specified, this option is ignored.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Command-line examples:

- Create new licenses using the **newlicense** mode with a different delimiter:

```
ada_util -u=user -p=pass -g=dba -m=newlicense
        -filename=TC_21910_004_newlicense_datafile.txt
```

The contents of the **TC_21910_004_newlicense_datafile.txt** shown below allows the use of the pipe character (|) in the entries (TC|21910_ITAR_1).

```
!$license_id$reason$type$citizenships
TC|21910_ITAR_1$To_Manage_Control_Over_Location$itar$US, JP, AL, AS
TC|21910_ITAR_2$To_Manage_Control_Over_Location$itar$US, JP, AL, AS
TC|21910_ITAR_3$To_Manage_Control_Over_Location$itar$US, JP, AL, AS
```

```
TC|21910_ITAR_4$To_Manage_Control_Over_Location$itar$US, JP, AL, AS
TC|21910_IP_4$To_Manage_Control_Over_Location$ip$US, JP
```

- To set a classification of **secret** on the **1234/A** UG master dataset attached to the item **1234** revision **A**:

```
ada_util -u=user -p=pass -g=group -m=setclassification
-classification=secret -Item_id=1234 -RevID=A
-Object-Type="UG Master" -Object-Name=1234/A -type=ip
```

- To set an IP clearance level of **secret** for the **user2** user:

```
ada_util -u=user -p=pass -g=group m=setclearance
-clearance=secret -userid=user2 -type=ip
```

- To create a new ITAR license **license001** with an expiration date of 11 December 2018 at 15:20:

```
ada_util -u=user -p=pass -g=group -m=newlicense
-license_id=license001 -date="11-dec-2018 15:20" -type=itar
```

- To create the **ITAR_license001** ITAR license with a category of Category A:

```
ada_util -u=user -p=pass -g=group -m=newlicense
-license_id=ITAR_license001 -category="Category A"
```

- To create a new **ITAR_license01** with an allowed citizenship for Great Britain:

```
ada_util -u=user -p=password -g=group -m=newlicense
-license_id=ITAR_license01 -type=itar -citizenships=GB
```

- To add **user2** user to the **license001** license:

```
ada_util -u=user -p=pass -g=group -m=adduser
-license_id=license001 -userid=user2
```

- To apply the **license001** license to the **1234/A** dataset attached to item **1234** revision **A**:

```
ada_util -u=user -p=pass -g=group -m=addlicense
-license_id=license001 -Item_id=1234 -RevID=A
-ObjectType="UG Master" -ObjectName=1234/A
```

- To apply the **license001** license to the **1234/A** dataset attached to item **1234** revision **A** without specifying the **-RevID**, **-ObjectType**, and **-Object-Name** options:

```
ada_util -u=user -p=pass -g=group -m=addlicense
        -license_id=license001 -Item_id=1234
```

- To update the **ITAR_license001** ITAR license to Category B:

```
ada_util -u=user -p=pass -g=group -m=modlicense
        -license_id=ITAR_license001 -category="Category B"
```

- To update user citizenships on the license **ITAR_license01** to Great Britain and Japan:

```
ada_util -u=user -p=password -g=group -m=modlicense
        -license_id=ITAR_license01 -citizenships=GB, JP
```

File-based examples:

The **ada_util** supports the use of a **puid** column in the input file to identify objects when using the **addlicense**, **removelicense**, and **setclassification** arguments. This is useful if the object's id contains special characters, which are not supported in the input file.

Example:

If you use the **setclassification** argument, you cannot use the item id in this case because it contains special characters:

```
!|item|classification|type
Model_8"x11"|top-secret|ip
```

Instead, identify the item using its puid.

```
!|puid|classification|type
w90AAEXS5ZdsKD|top-secret|ip
```

Use the pipe character (|) as a delimiter in your files and the pound sign (#) at the beginning of a file for comments.

- Create new licenses using the **newlicense** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=newlicense
        -filename=TC_21910_004_newlicense_datafile.txt
```

TC_21910_004_newlicense_datafile.txt contents:

```
!|license_id|reason|type|citizenships
TC21910_ITAR_1|To_Manage_Control_Over_Location|itar|US, JP, AL, AS
TC21910_ITAR_2|To_Manage_Control_Over_Location|itar|US, JP, AL, AS
```

```
TC21910_ITAR_3|To_Manage_Control_Over_Location|itar|US,JP,AL,AS
TC21910_ITAR_4|To_Manage_Control_Over_Location|itar|US,JP,AL,AS
TC21910_IP_4|To_Manage_Control_Over_Location|ip|US,JP
```

- Modify licenses using the **modlicense** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=modlicense
          -filename=TC_21910_104_modlicense_datafile.txt
```

TC_21910_104_modlicense_datafile.txt contents:

```
!|license_id|reason
TC21910_ITAR_1|To_Manage_Control_Over_Location_Modified
TC21910_ITAR_2|To_Manage_Control_Over_Location_Modified
TC21910_ITAR_3|To_Manage_Control_Over_Location_Modified
TC21910_ITAR_4|To_Manage_Control_Over_Location_Modified
TC21910_IP_4|To_Manage_Control_Over_Location_Modified
```

- Add licenses using the **addlicense** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=addlicense
          -filename=TC_21910_104_addlicense_datafile.txt
```

TC_21910_104_addlicense_datafile.txt contents:

```
!|license_id|Item_id|ead_paragraph
TC21910_ITAR_1|TC_21910_Item_4|ITAR_Section_127_1_1
TC21910_ITAR_2|TC_21910_Item_5|ITAR_Section_127_1_1
TC21910_ITAR_3|TC_21910_Item_6
TC21910_ITAR_4|TC_21910_Item_7
```

- Remove licenses using the **removelicense** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=removelicense
          -filename=TC_21910_304_removelicense_datafile.txt
```

TC_21910_304_removelicense_datafile.txt contents:

```
!|license_id|Item_id
TC21910_ITAR_1|TC_21910_Item_4
TC21910_ITAR_2|TC_21910_Item_5
```

```
TC21910_ITAR_3|TC_21910_Item_6
TC21910_ITAR_4|TC_21910_Item_7
```

- Add users using the **adduser** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=adduser
          -filename=TC_21910_404_adduser_datafile.txt
```

TC_21910_404_adduser_datafile.txt contents:

```
!|license_id|userid
TC21910_IP_4|TC_21910_User_4
TC21910_ITAR_1|TC_21910_User_5
TC21910_ITAR_2|TC_21910_User_6
TC21910_ITAR_3|TC_21910_User_7
TC21910_ITAR_2|TC_21910_User_7
```

- Remove users using the **removeuser** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=removeuser
          -filename=TC_21910_504_removeuser_datafile.txt
```

TC_21910_504_removeuser_datafile.txt contents:

```
!|license_id|userid
TC21910_IP_4|TC_21910_User_4
TC21910_ITAR_1|TC_21910_User_5
TC21910_ITAR_2|TC_21910_User_6
TC21910_ITAR_3|TC_21910_User_7
```

- Add groups using the **addgroup** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=addgroup
          -filename=TC_21910_604_addgroup_datafile.txt
```

TC_21910_604_addgroup_datafile.txt contents:

```
!|license_id|groupid
TC21910_IP_4|TC_21910_Group_4
TC21910_ITAR_1|TC_21910_Group_5
```

```
TC21910_ITAR_2|TC_21910_Group_6
TC21910_ITAR_3|TC_21910_Group_7
```

- Remove groups using the **removegroup** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=removegroup
          -filename=TC_21910_704_removegroup_datafile.txt
```

TC_21910_704_removegroup_datafile.txt contents:

```
!|license_id|groupid
TC21910_IP_4|TC_21910_Group_4
TC21910_ITAR_1|TC_21910_Group_5
TC21910_ITAR_2|TC_21910_Group_6
TC21910_ITAR_3|TC_21910_Group_7
```

- Set classification using the **setclassification** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=setclassification
          -filename=TC_21910_904_setclassification_datafile.txt
```

TC_21910_904_setclassification_datafile.txt contents:

```
!|item|classification|type
TC21910_Item_4|secret|ip
TC21910_Item_5|secret|itar
TC21910_Item_6|top-secret|itar
```

- Set clearance using the **setclearance** mode:

```
ada_util -u=Tc-admin-user -p=password -g=group -m=setclearance
          -filename=TC_21910_804_setclearance_datafile.txt
```

TC_21910_804_setclearance_datafile.txt contents:

```
!|userid|clearance|type
TC21910_User_4|secret|ip
TC21910_User_5|secret|itar
TC21910_User_6|top-secret|itar
```

- To create a new ITAR license **license001** with an expiration date of 11 December 2018 at 15:20:

```
ada_util -u=user -p=pass -g=group -m=newlicense  
-filename=input_file_name -report_file=report_file.report  
-repeat_file=repeat_file.rep  
-o=log_file.log license_id=license001  
-date="11-dec-2018 15:20" -type=itar
```

generate_ada_input

Allows you to create an input file for the **ada_util** utility to create new or modify existing ADA licenses on a new site. Using this input file, you can quickly create or modify multiple licenses.

Use the following naming conventions for the input file:

- When *creating* ADA licenses on a new site:

ada_input_file_for_create_licenses_date-stamp.txt

- When *modifying* existing ADA licenses on a new site:

ada_input_file_for_update_licenses_date-stamp.txt

SYNTAX

```
generate_ada_input [-u=user-id {-p=password | -pf=password-file} g=group]
{ -from_date=date -to_date=date | -period=whole-number-0-to-365
[ { -from_date=date | -to_date=date } ] }
[-outputFilePath=directory-path] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-from_date

Specifies the start date of the period using the format DD-MMM-YYYY HH:MM, for example, 28-OCT-2020 12:01.

-to_date

Specifies the end date of the period using the format DD-MMM-YYYY HH:MM, for example, 28-OCT-2020 12:01.

-period

Specifies the number of days in the period, either from the date provided in **-from_date** or the date provided in **-to_date**.

If no **-from_date** or **-to_date** is specified, then **-period** is calculated backward from today's date.

If both **-from_date** and **-to_date** are specified, then **-period** is calculated backward from **-to_date**.

-outputFilePath

(Optional) Specifies the directory location on the disk where the output file is created.

If this argument is not specified, the file is created in the directory in which you invoke the utility.

RULES

The following rules apply:

1. When you use **-from_date** alone, all licenses created after the **-from_date** until today's date are recorded.
2. When you use **-from_date -period**, all licenses created after the **-from_date** for the period specified are recorded.
3. When you use **-to_date -period**, all licenses created from the **-to_date** backward for the period specified are recorded.
4. When you use **-period** alone, all licenses created from today's date backward for the period specified are recorded.
5. When you use no **-from_date/-to_date/-period** arguments, no licenses are created, as one of these arguments must be specified.

6. When you use **-from_date** and **-to_date**, all licenses are created from the **-from_date** to the **-to_date**.
7. When you use **-from_date**, **-to_date**, and **-period**, all licenses created from today's date backward for the period specified are recorded.

EXAMPLES

1. Given today's date is 30-Jan-19 and an ITAR license, **itar1**, created on 15-Jan-19 is present and the following command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=group
-from_date="1-Jan-19 00:00"
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenship itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships
```

2. Given an ITAR license, **itar1**, created on 14-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba
-from_date="1-Jan-19 00:00" -period=20
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships
```

3. Given an ITAR license, **itar1**, created on 14-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba
                  -to_date="31-Jan-19 00:00" -period=20
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships
```

4. Given today's date is 31-Jan-19 and an ITAR license, **itar1**, created on 14-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba -period=20
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!!license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships
```

5. Given an ITAR license, **itar1**, created on 14-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba
                  -from_date="1-Jan-19 00:00" -to_date="15-Jan-19 23:59"
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenship
```

6. Given an ITAR license, **itar1**, created on 14-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba
-from_date="1-Jan-19 00:00" -to_date="15-Jan-19 23:59" -period=2
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

No license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships
```

7. Given today's date is 30-Jan-19 and an ITAR license, **itar1**, created on 15-Jan-19 and modified on 27-Jan-19 and another ITAR license, **itar1**, created on 26-Jan-19 and modified on 28-Jan-19 is present and following the command is executed:

```
generate_ada_input -u=Tc-admin-user -p=password -g=dba -period=5
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar2|||ITAR_License||Para-1|Category A|
```

One license is found and the contents of the **ada_input_file_for_create_licenses_date-stamp.txt** file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|
category|citizenships itar1|||ITAR_License||Para-1|Category A|
```

Generally, the content of the file is:

```
!|license_id|date|reason|type|lock_date|qualifying_cfr|category|citizenships
itar1|15-Aug-2019 00:00||ITAR_License||Para-1|Category A|
itar2|15-Aug-2019 00:00||ITAR_License||Para-2|Category B|GB, IN, US
ip1||reason 1|IP_License|15-Sep-2019 00:00||Category A|
ip2|15-Sep-2019 00:00|reason 2|IP_License||Category B|
ip3||reason 3|IP_License|15-Sep-2019 12:45||Category A|
```

where:

- The header is displayed as **!|license_id|date|reason|type|lock_date|qualifying_cfr|category|citizenships.**
- License data is displayed on every new line following the header.

install_am_privilege

Installs Access Manager privileges.

SYNTAX

```
install_am_privilege -u=user-id {-p=password | -pf=password-file} [-g=group]  
-n=privilege-name  
[-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-n

Specifies the privilege name.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Perform a custom detach:

```
install_am_privilege -u=Tc-admin-user -p=password  
-g=group -n=ADA_Custom_Detach
```

install_authorization_rules

Creates system-level administration authorization rules.

SYNTAX

```
install_authorization_rules [-u=user-id {-p=password | -pf=password-file} [-g=group]
-function {install | create | add | listaccessors | listapplications | listutilities}
[-name=application-or-utility-name]
[-ruledomain=rule-domain-value]
[-role=role-name]
[-group=group-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

function

Specifies one of the following functions:

install

Installs standard authorization rules for administration applications and utilities.

create

Creates new authorization rules. If you specify this function, you must include values for the **name** and **ruledomain** arguments.

add

Adds a new accessor to an existing authorization rule. If you specify this function, you must include values for the **name**, **ruledomain**, and **group** arguments.

listaccessors

Lists accessors specified by the **name** and **ruledomain** arguments.

listapplications

Lists all application names for which rules are defined in the database.

listutilities

Lists all utility names for which rules are defined in the database.

-name

Specifies the application or utility name. Use this argument with the **create**, **add**, or **listaccessors** functions.

-ruledomain

Specifies the value of the rule domain. Use this argument with the **create**, **add**, or **listaccessors** functions.

-role

Specifies the role name.

-group

Specifies the group name. Use this argument with the **add** function.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Install the default authorization rules for applications and utilities:

```
install_authorization_rules -u=Tc-admin-user -p=password
-g=dba -install
```

- Create authorization rule for the **APP_1** application:

```
install_authorization_rules -u=Tc-admin-user -p=password
-g=dba -create -name=APP_1 -ruledomain=application
```

- Add the **GRP_1** group as a valid accessor for the **APP_1** application:

```
install_authorization_rules -u=Tc-admin-user -p=password
-g=dba -add -name=APP_1 -ruledomain=application -group=GRP_1
```

- List all the accessors for the **APP_1** application:

```
install_authorization_rules -u=Tc-admin-user -p=password
-g=group -listaccessors -name=APP_1 -ruledomain=application
```

- List all application names for which rules are defined in the database:

```
install_authorization_rules -u=Tc-admin-user -p=password
-listapplications
```

- List all utility names for which rules are defined in the database:

```
install_authorization_rules -u=Tc-admin-user -p=password
-listutilities
```

install_callback

Registers the callbacks and persists them in the database.

SYNTAX

```
install_callback -u=user-ID {-p=password | -pf=password-file} [-g=group]
[-mode={install | create | modify | delete | list}]
[-type= type-of-callback]
[-library=library-name]
[-function=function-name]
[-name=callback-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-mode

Specifies the callback mode.

- **install**

Specifies the install mode.

- **create**

Specifies the create mode.

If running in this mode, specify **-type**, **-library**, **-function**, or **-name**.

- **modify**

Specifies the modify mode.

If running in this mode, specify **-type**, **-library**, **-function**, or **-name**.

- **delete**

Specifies the delete mode.

If running in this mode, specify **-type** or **-name**.

- **list**

Specifies the list mode.

-type

Specifies the type for the **create**, **modify**, or **delete** mode.

-library

Specifies the library for the **create** or **modify** mode.

-function

Specifies the function for the **create** or **modify** mode.

-name

Specifies the name for the **create**, **modify**, or **delete** mode.

-h

Displays help for this utility.

install_vminfo_acl

Creates access control list (ACL) rules for migration of the existing Teamcenter volume files. The Volume Management application introduces changes to the Access Manager (AM) rule tree. This utility verifies whether the AM rule tree contains the required rules (**HSM_Info** and **VM_Info**). If not, the rules are added to inherit the access privileges of the named ACL **POM Open Access** in par with the **ImanFile** object and saves the changes.

This utility runs automatically at install. Typically, there is no need for administrators to run the utility again. In cases where an administrator has overwritten the rule tree with custom rules, this utility can be run to ensure the required rules are added to the rule tree.

SYNTAX

```
install_vminfo_acl -u=user-id {-p=password | -pf=password-file} [-g=group] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-v

Verbose mode. Provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility is intended only for system-level users.

RETURN VALUES

Return value 0
upon success

Return value 1
upon failure

EXAMPLES

```
install_vminfo -u=Tc-admin-user -p=password -g=group -v
```

Business Modeler IDE

bmide_admin_data

Create the administration data folders and script files in a BMIDE template project.

Syntax

```
bmide_admin_data  
-createProject  
-inputFile=input-file-path
```

```
bmide_admin_data
-generatePackage
-projectLocation=project-file-path
-version=custom-software-version
-buildVersion=custom-bulid-version
```

Arguments

-createProject

Sets the utility to create a proeject using an input file.

-inputFile

Specify the location of the **.properties** input file containing the information about the project.

-generatePackage

Add the admin data folders and script files to an existing project.

-projectLocation

Specifies the file path of the new project.

-version

Specifies the version of the new project.

-buildVersion (optional)

Specifies the build version of the new project.

Environment

As specified in [Setting up a Teamcenter command-line environment](#).

Files

Input file containing the creation information about the project.

Example:

project.properties

```
application_id=a9101
description=Description
applicationId_prefix=A9
nature=external
workspace=C:\\admin_data\\workspace\\test
```

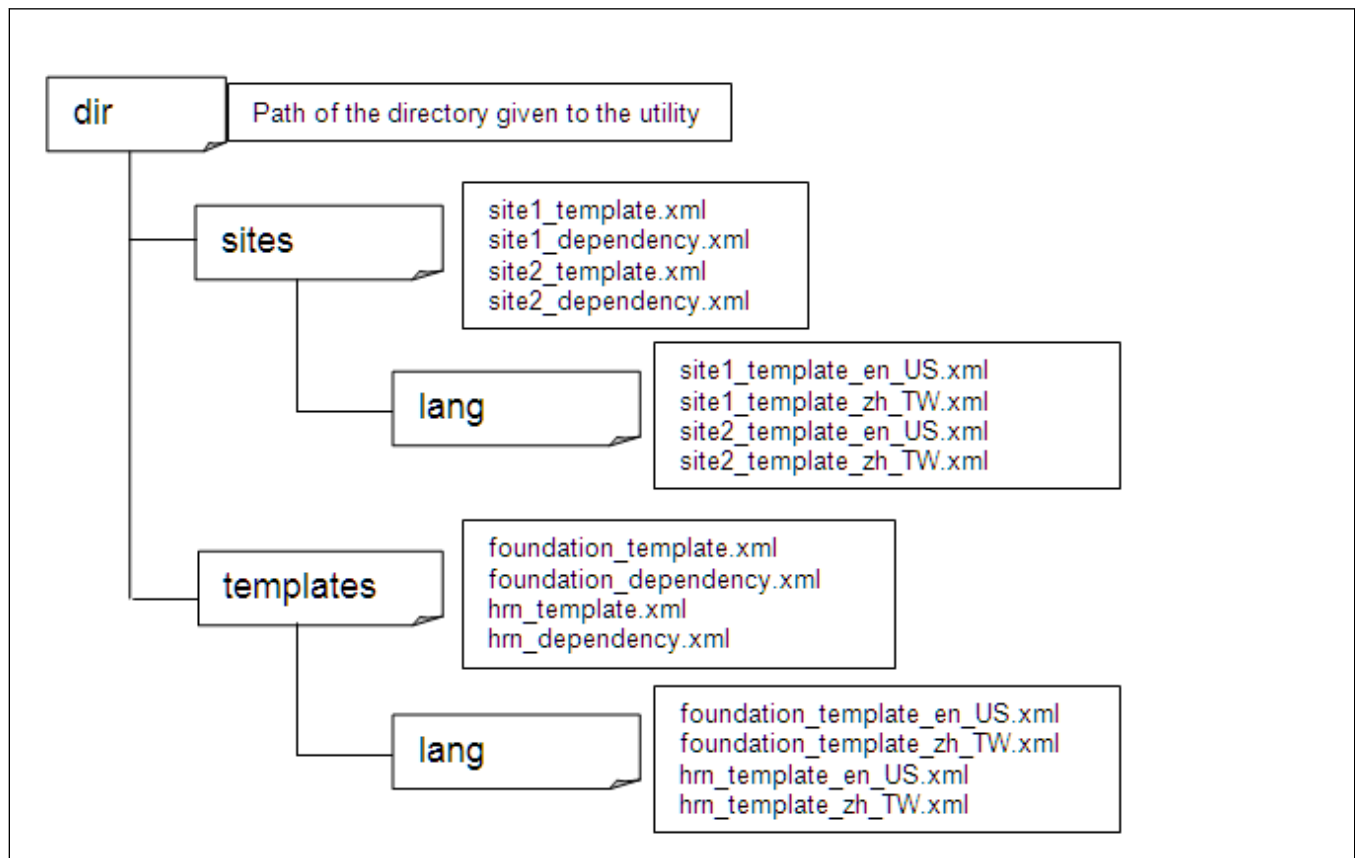
Restrictions

None.

bmide_commontemplategenerator

Generates a common template between two or more sites and generates site-specific templates for each site. Use this utility when you have multiple sites containing custom data model. Previously, you had to manually analyze the site templates generated at each site to create a common template. This utility eliminates the manual work to create a template that contains custom data model common to all sites.

Place template files in the following directory structure:



Directory structure required by the bmide_commontemplategenerator utility

Obtain the files to place into the directories from the packaged *template-name_template.zip* files. The **sites** directory contains the templates from the different sites, the **templates** directory contains the standard templates that the site templates are dependent upon, and the **lang** directories contain the localization files used by the templates.

SYNTAX

bmide_commontemplategenerator

-dir=site-templates-directory
 -name=common-template-name
 -displayname=common-template-display-name

-outputdir=*output-directory*
[-h]

ARGUMENTS

-dir

Specifies the path of the directory where sites and templates folders are located. The directory should contain a **sites** subdirectory that contains all the site templates and dependency files and a **templates** subdirectory that contains all the dependent templates. The **sites** and **templates** directories should contain a **lang** subdirectory containing locale files.

-name

Specifies the name of the common template to be generated.

-displayname

Specifies the display name of the common template to be generated.

-outputdir

Specifies the path of the directory where the generated files are to be placed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

This utility has a requirement that a minimum of 1500 MB memory must be allocated to the Java heap space. Before running the utility, set the following environment variable:

```
set BMIDE_SCRIPT_ARGS=-Xmx1500M
```

EXAMPLES

```
bmide_commontemplategenerator -dir=c:\templates\temp
-name=commontemplate -displayname="Common Template"
-outputdir=c:\templates\temp\output
```

bmide_comparator

Compares two complete Teamcenter model files and generates a differences file. This utility must be run with either the **-schema** or **-all** argument.

SYNTAX

```
bmide_comparator -compare={schema | all} -old=old-model-file-path  
-new=new-model-file-path -delta=differences-file-path  
[-log=log-file-path] [-h]
```

ARGUMENTS

-compare={schema | all}

Compare data model. You must specify one of these options:

- **schema**

Compares only classes.

- **all**

Compares all elements.

-old

Specifies the file path and name of the file containing the old Teamcenter model.

-new

Specifies the file path and name of the file containing the new Teamcenter model.

-delta

Specifies the file path and name of the file into which data model differences will be written.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_consolidator

Consolidates all templates listed in the master file into a single file.

SYNTAX

```
bmide_consolidator -dir=master-file-directory -file=path-for-consolidated-file  
-consolidate=[all | template | locale]  
-forceconsolidate  
[-version=version]  
[-h]
```

ARGUMENTS

-consolidate

all

Consolidates the template and its localization files.

template

Consolidates templates.

locale

Consolidates template localization files.

-dir

Specifies the directory path and name of the directory containing the **master.xml** file and the list of template files to be consolidated.

-file

Specifies the file path and name of the file which will contain the consolidated Teamcenter model.

-forceconsolidate

Indicates that localization files must be consolidated irrespective of what is included in the **master.xml** file.

-version

Specifies version for the consolidated file. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

RESTRICTIONS

None.

bmide_deployment_lock

Prevents simultaneous deployments to a database from any source, including Teamcenter Environment Manager (TEM) and the Business Modeler IDE. Only administrative users are allowed to run this utility.

SYNTAX

```
bmide_deployment_lock -u=user-id {-p=password | -pf=password-file} [-g=group]
-lock | -release | -query
[-log=path]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-lock

Sets the deployment lock. If one already exists, the utility returns an error indicating a deployment is in process. If one does not exist, the utility sets the deployment lock and returns success.

-release

Releases the deployment lock. If one already exists, the utility removes the deployment lock and returns success. If one does not exist, the utility returns success.

-query

Queries for the deployment lock. If one already exists, a **-1** is returned. If one does not exist, the utility returns success.

-log

Specifies the full path to the log file containing utility execution results.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_generatecode

Autogenerates C/C++ code for business objects and property operations from the Business Modeler IDE.

SYNTAX

bmide_generatecode

-**templateProj**=*source-template-project-input-location*
 -**templateDeps**=*dependent-templates-input-location*
 -**srcDir**=*skeleton-implementation-classes-output-location*
 -**gensrcDir**=*autogenerated-classes-output-location*

-**serviceLibs**=[**all** | *service-libraries*]
 -**log**=*log-output-file*
 [-h]

ARGUMENTS

-templateProj

Specifies the file path and name of the input location of the source template project.

-templateDeps

Specifies the file path and name of the input location of the dependent templates.

-srcDir

Specifies the file path and name of the output location of the skeleton implementation classes. This argument is optional.

-gensrcDir

Specifies the file path and name of the output location for autogenerated classes.

-serviceLibs

Generates code for all custom service libraries defined in the specified Business Modeler IDE project. Specify a comma-delimited list of service library names or **all**, for example, **-serviceLibs=all**.

Before running this utility with the **-serviceLibs** argument, you must do the following:

1. Set the **JDK_HOME** environment variable to point to the installed JDK location.
2. Add the **JDK_HOME\bin** directory to the path variable.
3. In the Business Modeler IDE project, ensure that the **SoaExternalBuild.SoaClientKitLocation** value in the **ProjectInfo.xml** file is set to the location of the *soa_client* folder. This corresponds to the value in the **Teamcenter Services client kit home** box in the **Build Configuration** dialog box for the project. To access this dialog box, in the **Navigator** view,

right-click the project and choose **Properties**, and in the left pane, choose **Teamcenter**→**Build configuration**.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
bmide_generatecode
  -templateProj=D:\udu\meta_dev10\custom1
  -templateDeps=D:\udu\meta_dev10\templates
  -srcDir=D D:\udu\meta_dev10\custom1\src\server
  -gensrcDir=D:\udu\meta_dev10\custom1\output\server\gensrc
  -log=D:\udu\meta_dev10\CodeGenUtil.log
```

bmide_generate_compare_report

Reports the differences between two sets of data models. For example, you can compare two Teamcenter database sites to determine if the data model is the same in both sites. The report generated by this utility shows the differences. You must provide two consolidated data model sources as input to generate the report. (You cannot provide other source for input.)

You can also generate this report using the Business Modeler IDE.

For more information, see *Run data model reports* in *BMIDE for Data Model Design*.

SYNTAX

bmide_generate_compare_report

```
-file=consolidated-model-file
-comparefile=consolidated-model-file-to-compare
-reportfile=generated-report-file
-showequal=[true | false]
-log=log-file
[-h]
```

ARGUMENTS

-file

Specifies the initial consolidated model file used to generate the report. A consolidated model file can be obtained by running the **business_model_extractor** utility to extract all the elements or by getting the **model.xml** file from the *TC_DATA\model* directory of a Teamcenter database site.

-comparefile

Specifies the consolidated model file to be compared to the initial file specified in the **-file** argument.

-reportfile

Specifies the file where the report is generated.

-showequal

Defines whether equal values from both data models are shown in the generated report. The valid values are **true** or **false**. The default value is **false**. If the value is **true**, the equal attributes are shown in the generated report.

-log

Specifies the path of the log file containing results of the execution.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_generate_condition_report

Reports the details of a condition and all model elements that use it in a set of data model. Conditions are attached to various business rules, LOVS, and so on to define the behavior of Teamcenter. The generated report is a single HTML page showing the model elements that refer to the condition. To generate this report, you must provide a condition name and a consolidated model file for the data model source.

You can also generate this report using the Business Modeler IDE.

For more information, see *Run data model reports*.

SYNTAX

bmide_generate_condition_report

-file=consolidated-model-file
-condition=condition-name
-reportfile=generated-report-file
-log=log-file
[-h]

ARGUMENTS

-file

Specifies the consolidated model file used to generate the report. A consolidated model file can be obtained by running the **business_model_extractor** utility to extract all the elements or by getting the **model.xml** file from the *TC_DATA\model* directory of a Teamcenter database site.

-condition

Specifies the condition name for which the report is to be generated.

-reportfile

Specifies the file where the report is generated.

-log

Specifies the path of the log file containing results of the execution.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_generate_datamodel_report

Reports the details of a given category of model elements.

Business Modeler IDE users can define a number of model elements and store them in a Business Modeler IDE template. These templates can be deployed to any Teamcenter database. At times a Business Modeler IDE user may be interested in generating a report of all LOVS or GRM rules in the system, or a report that shows a combination of multiple model element categories such as all deep copy rules and all naming rules. A user can use the Business Modeler IDE client to examine all of this information; however, this report offers an easier means to examine all elements within a given category by generating this information into a single HTML page.

The data model source for the report can be specified only in the form of a consolidated model file, for example, `TC_DATA\model\model.xml`. (You cannot provide other source for input.)

You can also generate this report using the Business Modeler IDE.

For more information, see *Run data model reports in BMIDE for Data Model Design*.

Note:

Not specifying the **-element** argument for the data model report generates a report for all elements. Because all elements are generated into a single file, this report has limitations. As the number of categories and elements within a category grows, so does the resulting file size. In cases where you want to generate a report of multiple categories, try using the **bmide_generate_datamodel_doc_report** utility.

SYNTAX

bmide_generate_datamodel_report

```
-file=consolidated-model-file
-element=element-name
-reportfile=generated-report-file
-log=log-file
[-h]
```

ARGUMENTS

-file

Specifies the initial consolidated model file used to generate the report. A consolidated model file can be obtained by running the **business_model_extractor** utility to extract all the elements or by getting the **model.xml** file from the `TC_DATA\model` directory of a Teamcenter database site.

-element

Specifies the comma-separated element names to include in report.

-reportfile

Specifies the file where the report is generated.

-log

Specifies the path of the log file containing results of the execution.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This report only generates business objects and classes that are under the **Item**, **ItemRevision**, **Form**, **Dataset**, and **Folder** business objects or classes. This report also generates classes that are the form storage classes of any item master form or item revision master form business object. All other classes and business objects are not shown as they are considered internal to the product. Therefore, if you supply your own template in the target directory, only the business objects and classes that fall into these categories are shown in the report.

EXAMPLES

- To generate a report of all lists of values (LOVs), use the following command:

```
bmid_generate_datamodel_report -file=model-file-path-and-name -element=lov
-reportfile=report-file-path-and-name.html
```

For example:

```
bmid_generate_datamodel_report
-file=D:\apps\Siemens\tcdata\model\model.xml -element=lov
-reportfile=D:\downloads\data_model_report_lovs.html
```

- To generate a complete list of element names that are supported as arguments, use the following command:

```
bmid_generate_datamodel_report -element -h
```


bmide_generate_datamodel_doc_report

Generates an HTML data model report of business objects, business rules, and LOVs of the current Teamcenter version. The data model for the report is built using the Business Modeler IDE templates found in the target templates directory specified by the user (for example, *TC_ROOT/bmide/templates*).

You can also generate this report using the Business Modeler IDE.

For more information, see *Run data model reports* in *BMIDE for Data Model Design*.

The data model report consists of the following sections:

- **Overview**

Provides an overview of the data model report.

- **Template Data Model**

Provides a list of the template names that contribute to the report.

- **What's New**

Provides a list of what has changed in the data model since a previous version.

- **Glossary**

Provides a glossary of all the data model elements managed by the Business Modeler IDE.

- **Deprecated**

Provides a list of all libraries and operations that are deprecated.

SYNTAX

```
bmide_generate_datamodel_doc_report
-targetTemplatesDir=target-file-path-name
[-sourceReleaseVersions=[Tcrelease | all]
-outputDir=directory-for-generated-reports
[-skip=XML-file-excluded-templates]
[-h]
```

ARGUMENTS

-targetTemplatesDir

Specifies the full path to the directory containing all templates from the target version for which the data model report must be generated, such as `TC_DATA/model` or `TC_ROOT/bmide/templates`.

-sourceReleaseVersions

Specifies a comma-separated list of Teamcenter versions to generate a comparison, for example, **-sourceReleaseVersions=Tc820,Tc830**. For a complete list of the Teamcenter versions to use with the argument, use the **-h** argument to see the help text for the utility.

To generate a report for all versions, specify **-sourceReleaseVersions=all**.

-outputDir

Specifies the full path to the directory where the reports must be generated.

-skip

Specifies a list of templates in the target templates directory to skip during processing. This argument should specify the full path to the XML file that has the list of templates. Following is an example of the XML file format:

```
<?xml version="1.0" encoding="UTF-8" standalone="no">
<TcBusinessdataIncludes>
  <exclude file="hrn_template.xml"/>
  <exclude file="erp_template.xml"/>
</TcBusinessdataIncludes>
```

-h

Displays help for this utility.

RESTRICTIONS

Following are the limitations of this utility:

- This report only generates business objects and classes that are under the **Item**, **ItemRevision**, **Form**, **Dataset**, and **Folder** business objects or classes. This report also generates classes that are the form storage classes of any item master form or item revision master form business object. All other classes and business objects are not shown as they are considered internal to the product. Therefore, if you supply your own template in the target directory, only the business objects and classes that fall into these categories are shown in the report.
- The reporting tool contains a record of all Teamcenter templates that are included with Teamcenter for each prior version. Therefore, when generating the **What's New** section, the report is able to accurately generate a comparison of all elements in a target version against a designated former source version. This results in the **What's New** comparison displaying new, changed, and removed for each element. Because the reporting tool does not include a record of any other templates, including third-party and customer templates, the report automatically lists all elements from these templates as new in the **What's New** section. Disregard the new status because it does not reflect whether the element was really new, changed, or removed.

EXAMPLES

- To generate an HTML data model report, enter a command like the following on a single line:

```
bmide_generate_datamodel_doc_report
-targetTemplatesDir=C:\Siemens\config1\tcdata\model
-sourceReleaseVersions=all -outputDir=C:\datamodelreports\tc9.0
-skip=C:\skiplist.xml
```

- To generate the **What's New** section of the report, choose one or more source versions to compare with the current version, as shown in the following examples:

- To generate a report that compares the data model, use the following command:

```
bmide_generate_datamodel_doc_report
-targetTemplatesDir=C:\Siemens\config1\tcdata\model
-sourceReleaseVersions=Tc810
-outputDir=C:\datamodelreports\tc9.0
```

- To skip templates during generation, provide a list of templates in a file. For example, to skip the **hrn_template** and **erp_template** templates, use the following **skiplist.xml** file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TcBusinessDataIncludes>
  <exclude file="hrn_template.xml"/>
  <exclude file="erp_template.xml"/>
</TcBusinessDataIncludes>
```

To use the **skiplist.xml** file, run the following command:

```
bmide_generate_datamodel_doc_report
-targetTemplatesDir=C:\Siemens\config1\tcdata\model
-sourceReleaseVersions=all
-outputDir=C:\datamodelreports\tc9.0
-skip=C:\skiplist.xml
```

If a template is skipped, all the templates dependent on it are also be skipped. For example, if **template2** is dependent on **template1**, and **template1** is specified in the skip list file, **template1** and **template2** are both be skipped in the generation of the report.

bmide_generate_package

Packages a Business Modeler IDE template project or composite template project for distribution. The utility is particularly useful in situations where frequent scheduled builds are desired. The utility does not require that Business Modeler IDE is running.

The utility generates the same package that you can generate from within Business Modeler IDE. See *Generate a software package for distribution in BMIDE for Data Model Design*.

USAGE

```
bmide_generate_package
-projectLocation=<Location of the project which is to be packaged>
[-packageLocation=<Location where the package is to be created>]
-dependencyTemplateFolder=<Location of the dependency templates>
[-codeGenerationFolder=<Location of the project's compiled generated
code>]
-softwareVersion=<Software Version for the custom software package>
[-buildVersion=<Build Version for the custom software package>]
[-allPlatform]
[-log=<Log file>]
[-h]
```

ARGUMENTS

-projectLocation

Location of the BMIDE project which needs to be packaged.

-packageLocation

Location where the package is to be created. This is an optional argument. Default location is **<project_loc>\output**

-dependencyTemplateFolder

Location of the dependent templates needed for the BMIDE project.

-codeGenerationFolder

Location of the project's compiled generated code (i.e., <project>\output\wntx64). This argument is required only if the project contains generated Meta Model or SOA code.

-softwareVersion

Software Version for the generated software package. This is an optional argument. The value provided for this argument must follow one of the following patterns:

Pattern: int	Examples: 1, 2, 3
Pattern: int.int	Examples: 1.1, 1.2, 2.3
Pattern: int.int.int	Examples: 1.1.1, 1.2.1, 2.3.1

Pattern: `int.int.int.int` Examples: `1.1.1.2`, `1.2.1.1`, `2.3.1.1`

-buildVersion

Internal Build Version for the generated software package. This is an optional argument. The value provided for this argument must follow one of the following patterns:

Pattern: `int` Examples: `1`, `2`, `3`

Pattern: `int.int` Examples: `1.1`, `1.2`, `2.3`

Pattern: `int.int.int` Examples: `1.1.1`, `1.2.1`, `2.3.1`

Pattern: `int.int.int.int` Examples: `1.1.1.2`, `1.2.1.1`, `2.3.1.1`

-allPlatform

Generates a software package that can be deployed to a Teamcenter business logic server on any platform (wntx64 or lnx64). This is useful when your software package only has a BMIDE template and no platform specific files such as compiled libraries that are specific to an OS. Note that if your BMIDE template project is configured to build and include the `rt_server.zip` with the generated software, the BMIDE will not build these into the generated software package to ensure that the software package can be installed to either wntx64 or lnx64.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_instance_delete

Reports and optionally deletes instances for data model elements to support template removal processing. Administrator access is required to run this utility.

SYNTAX

```
bmide_instance_delete -u=user-id {-p=password | -pf=password-file} [-g=group]
-template=template -mode=[dryrun | count | export | delete]
-instances=business-object-list
-file=delta-xml-file-path
-tcxmlOutput=tcxml-export-data-file-path
-listSupportedElements
[-log=log-file-path]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies one of the following operation modes:

- **dryrun** (default)

Reports all instances of the specified input including referencers of each instance. You can use the optional **limit** argument can be used with this option to limit the number of instances reported by the utility.

- **count**

Provides a count (only) of instances of the specified input.

- **export**

Provides TC XML export data file for instance backup and a report of all instances of the specified input including referencers of each instance.

- **delete**

Attempts to delete instances (along with their referencers) and report the results of the delete activity. Reports the same detail reporting as the **export** mode.

-limit

Specifies the number of instances to include in the report. The default is **100**. Specify **all** to include all instances in the report. The total instance count is always included in the report.

-instances

Specifies the type of instances to include in the report. This argument is mutually exclusive with the **-file** argument.

The argument format is *type-qualifier-name:model-element-name*. Single or multiple values (comma separated) are permitted. For example:

```
-instances="Type:MyItem"
```

```
-instances="Type:MyItem, Type:YourItem"
```

- The **type-qualifier-name** options are available from the **-listSupportedElements** argument.
- The **model-element-name** option must be a business object name for the specified **type-qualifier-name**.

-file

Specifies the location of the input delta transaction file. This argument is mutually exclusive with the **-instances** argument.

-tcxmlOutput

Specifies the location of the TC XML export data file used to back up instances to be deleted. Required with **export** mode and optional for **dryrun** and **delete** modes. It must be a valid XML file.

-listSupportedElements

Displays the current list of types supported by this utility.

-log

Specifies file path and name of the log file report for the utility. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To delete business object **MyItem** type instances:

```
bmid_instance_delete -u=username -p=password -g=dba
-mode=delete -tcxmlOutput=<filename> -instances="Type:MyItem"
```

- To report instances with business object type **MyItem** and **YourItem**:

```
bmid_instance_delete -u=username -p=password -g=dba
-mode=dryrun -instances="Type:MyItem,Type:YourItem"
```

- To delete **NoteType** business object instances with the name **MyNoteType1**:

```
bmid_instance_delete -u=username -p=password -g=dba
-mode=delete -tcxmlOutput=<filename> -instances="NoteType:MyNoteType1"
```

- To report **Tool** business object instances with the name **MyTool**:


```
bmide_instance_delete -u=username -p=password -g=dba
-mode=dryrun -instances="Tool:MyTool"
```

- To delete business object **MyForm** type instances:

```
bmide_instance_delete -u=username -p=password -g=dba
-mode=delete -tcxmlOutput=<filename> -instances="Type:MyForm"
```

- To delete business object **MyDataset** type instances:

```
bmide_instance_delete -u=username -p=password -g=dba
-mode=dryrun -instances="Type:MyDataset"
```

- To delete **MyItem** type instances, **NoteType** instances with the name **MyNoteType1**, and **Tool** instances with the name **MyTool**:

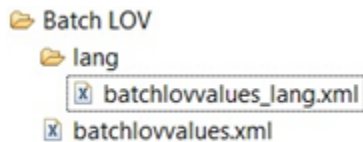
```
bmide_instance_delete -u=username -p=password -g=dba
-mode=delete -tcxmlOutput=<filename>
-instances="Type:MyItem,NoteType:MyNoteType1,Tool=MyTool"
```

bmide_manage_batch_lovs

Reports, extracts, and updates values for externally managed Lists of Values (LOVs).

Extracted batch LOVs comprise at least two files: the list definition, and localizations of list values.

When you use the **bmide_manage_batch_lovs** utility to update externally managed LOVs in the database, you submit the LOV list definition as a single XML file, with localizations in a **lang** subdirectory.



A consolidated localization file can contain localizations for multiple languages and is named **LOV_file-name_lang.xml**.



Individual locale files each contain localizations for one specific locale, and are named **LOV_file-name_lang_locale.xml**. For example, **file-name_en_US.xml**, **file-name_zh_CN.xml**.

You can place either a single consolidated file or multiple locale-specific files in the **lang** folder, but not both at the same time.

If you are using client cache at your site, then after updating the externally managed LOVs through this utility, to update the LOV cache stored on the server run the **generate_client_meta_cache** utility with the **generate lovs** command.

For information about externally managing LOVs, see *Create an externally managed (batch) list of values* in *BMIDE for Data Model Design*.

SYNTAX

```
bmide_manage_batch_lovs -u=user-id {-p=password | -pf=password-file} [-g=group]
-option=[update | extract | report]
-lov=[all | comma-separated-list-of-lov-names]
-file=file-path
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option

Valid values for option are:

- **update**

Updates the values for all the externally managed LOVs defined in the input file.

- **extract**

Extracts all the externally managed LOVs and their corresponding values into the specified file. The localizations also get extracted and are stored in the **lang** directory. Use with the **-lov** argument to specify the externally managed LOVs to be extracted.

- **report**

Provides a report of all the externally managed LOVs in the Teamcenter database. Use with the **-lov** argument to specify the LOVs in the report. The report is generated in the HTML format.

-lov

Use only in conjunction with the **-extract** or **-report** arguments. Valid values for option are:

- **all**

Extracts or runs a report on all externally managed LOVs.

- *comma-separated-list-of-lov-names*

Extracts or runs a report only on the listed externally managed LOVs.

If a list of LOV names is not specified, the **all** argument is assumed by default.

-file

Specifies the path to the file containing LOV values (when updating LOV values in the database), the path of file where the extracted LOVs are to be saved (when extracting LOV values from the database), or the file where the generated report needs to be saved (when running a report of LOV values in the database).

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- When you install the Business Modeler IDE, sample files are installed to `TC_ROOT\bmide\client\samples\externallymanagedlovs`.
- To extract all the LOV values, sub-LOV attachments, and localizations from the database, enter a command like the following on a single line:

```
bmide_manage_batch_lovs -u=username -p=password -g=dba
-option=extract -file=BatchLOV_LOV1_extracted.xml
```

Do this when you want to check the current values on any of the LOVs or to use the extracted file as the basis for the next set of changes if the original source input file is no longer available.

- To extract LOV values, sub-LOV attachments, and localizations for specific LOVs in the database, enter a command like the following on a single line:

```
bmid_manage_batch_lovs -u=username -p=password -g=dba
-option=extract -lov=BatchLOV_LOV1,BatchLOV_LOV2
-file=BatchLOV_extracted.xml
```

- To update the LOV values and localizations from an external file to the database, enter a command like the following on a single line:

```
bmid_manage_batch_lovs.bat -u=username -p=password -g=dba
-option=update -file=BatchLOV_LOV1.xml
```

Following is the **BatchLOV_LOV1.xml** file used in the example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<TcBusinessData xmlns="http://teamcenter.com/BusinessModel/TcBusinessData"
batchXSDVersion="1.0" >
<Change>
  <TcLOV description="Simple Batch LOV" lovType="ListOfValuesString"
name="Ab0LOV" usage="Exhaustive" isManagedExternally="true" >
    <TcLOVValue conditionName="isTrue" description="L1 Desc" value="L1"/>
    <TcLOVValue conditionName="isTrue" description="L2 Desc" value="L2"/>
    <TcLOVValue conditionName="isTrue" description="L3 Desc" value="L3"/>
    <TcLOVValue conditionName="isTrue" description="L4 Desc" value="L4"/>
    <TcLOVValue conditionName="isTrue" description="L5 Desc" value="L5"/>
    <TcLOVValue conditionName="isTrue" description="L6 Desc" value="L6"/>
    <TcLOVValue conditionName="isTrue" description="L7 Desc" value="L7"/>
    <TcLOVValue conditionName="isTrue" description="L8 Desc" value="L8"/>
    <TcLOVValue conditionName="isTrue" description="L9 Desc" value="L9"/>
    <TcLOVValue conditionName="isTrue" description="L10 Desc" value="L10"/>
  </TcLOV>
</Change>
</TcBusinessData>
```

Following is the **BatchLOV_LOV1_lang.xml** file that supplies the localization for the LOV values in the **BatchLOV_LOV1.xml** file. When you create a localization file, give it the same name as the LOV values file with **_lang** added to the file name, and place it in a **lang** subdirectory.

```
<?xml version="1.0" encoding="UTF-8"?>
<TcBusinessDataLocalization
xmlns="http://teamcenter.com/BusinessModel/TcBusinessDataLocalization"
batchXSDVersion="1.0" >
<Add>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L1" >Level 1</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L2" >Level 2</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L3" >Level 3</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L4" >Level 4</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L5" >Level 5</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L6" >Level 6</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L7" >Level 7</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L8" >Level 8</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L9" >Level 9</key>
  <key locale="en_US" id="LOVValue{::}Ab0LOV{::}L10" >Level 10</key>

  <key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}0" >Level 1 Desc</key>
  <key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}1" >Level 2 Desc</key>
```

```

<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}2" >Level 3 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}3" >Level 4 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}4" >Level 5 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}5" >Level 6 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}6" >Level 7 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}7" >Level 8 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}8" >Level 9 Desc</key>
<key locale="en_US" id="LOVValueDescription{::}Ab0LOV{::}9" >Level 10 esc</key>

</Add>
</TcBusinessDataLocalization>

```

- To add LOV values and sub-LOV attachments to cascading LOVs, enter a command like the following on a single line:

```

bmide_manage_batch_lovs.bat -u=username -p=password -g=dba
-option=update -file=BatchLOV_ValidCascading.xml

```

Note:

You can add sub-LOV attachments in this way only on externally managed LOVs.

Following is the **BatchLOV_ValidCascading.xml** file used in the example:

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<TcBusinessData xmlns="http://teamcenter.com/BusinessModel/TcBusinessData"
batchXSDVersion="1.0" >
<Add>
  <TcLOVValueSubLOVAttach conditionName="isTrue" subLOVName="ExcelTemplateRules"
targetLOVName="SampeCascadingBatchLOV" targetValue="v1"/>
  <TcLOVValueSubLOVAttach conditionName="isTrue" subLOVName="SampleBatchLOV"
targetLOVName="SampeCascadingBatchLOV" targetValue="v2"/>
</Add>
<Change>
  <TcLOV description="Sample Cascading Batch LOV" lovType="ListOfValuesString"
name="SampeCascadingBatchLOV" usage="Exhaustive" isManagedExternally="true">
    <TcLOVValue description="v1 description" value="v1"/>
    <TcLOVValue description="v2 description" value="v2"/>
  </TcLOV>

  <TcLOV description="Sample Batch LOV" lovType="ListOfValueString"
name="SampeBatchLOV" usage="Suggestive" isManagedExternally="true">
    <TcLOVValue description="sample" value="Sweden"/>
    <TcLOVValue description="sample" value="India"/>
    <TcLOVValue description="sample" value="UK"/>
    <TcLOVValue description="sample" value="USA"/>
    <TcLOVValue description="sample" value="Germany"/>
    <TcLOVValue description="sample" value="South Africa"/>
    <TcLOVValue description="sample" value="Australia"/>
    <TcLOVValue description="sample" value="New Zealand"/>
    <TcLOVValue description="sample" value="UAE"/>
    <TcLOVValue description="sample" value="Pakistan"/>
  </TcLOV>
</Change>
</TcBusinessData>

```

bmide_manage_templates

Adds solution templates to the database table.

SYNTAX

```
bmide_manage_templates -u=user-id {-p=password | pf=password-file} [-g=group]  
-option=option-type -templates=template-names [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option

Specifies whether to add templates to the database or list the existing templates in the database. Valid values are **add** or **list**.

-templates

Specifies the name of the template to add to the database. Multiple templates can be added by separating the names by a comma.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_modeltool

The **bmide_modeltool** utility loads the Business Modeler IDE data model and provides the model to other tools for consumption.

Since any install, upgrade, or deployment through TEM results in deployment of a Business Modeler IDE template, the system runs the **bmide_modeltool** utility as part of the TEM operation. When Live Update is used to make changes, such as to modify a constant, it may be necessary to manually run **bmide_modeltool**.

SYNTAX

```
bmide_modeltool -u=username {-p=password | -pf=password-file} [-g=group] -tool={all | tool-name}
[-mode={install | upgrade | update}] -target_dir=target-directory [-model_file=model-file-name] [-log=log-file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user password. This argument is mutually exclusive with the **-pf** argument.

Note:

The password must not contain any of the following characters:

! @ \$ % = & ' " ; , . < > () { } "

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-tool

The name of the tool to open. If **all** is entered, all tools are opened. (If the **mode** parameter is specified, then only tools that are supposed to run during the specified mode run.) Valid options for the tool name include the following:

tc_ace_solr_schema_gen

tc_productscope_solr_schema_gen

tc_ace_to_solr_xslt_gen

smart_discovery_solr_xslt_gen

smart_discovery_xsd_gen

occmgmt4bom_indexer_transfermode_gen

smart_discovery_transfer_mode_generator

tc_solr_schema_gen

tc_to_solr_xslt_gen

lowlevel_tcxml_xsd_gen

tcftsindexer_transfermode_gen

tc_clientcache_gen

Generates the Client Cache and uploads it to the database.

tc_entcba_ebom_types_pref_updater

Updates the Part, Design and ProductEBOM type preferences based upon business constant FndOPartDesignQualifier.

tc_clipsrules_gen

Generates the CLIPS rule file and upload it to the database.

tc_tcplmxmlschema_gen

Generates the low level TCXML schema file.

-mode

Specifies the mode or context for the tool operation.

install - Only tools that are supposed to run during installation run.

upgrade - Only tools that are supposed to run during upgrade run.

update - Only tools that are supposed to run during update run.

-target_dir

The location for output files.

-model_file

If a valid XML file is provided via the "model_file" parameter, then this XML file is used to generate the BMIDE data model.

If this parameter is not provided, then the data model is extracted from the database and used to generate the BMIDE data model.

-log

Specifies the name of the log file for the operation.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

RESTRICTIONS

None.

EXAMPLES

- To update tools after changes were deployed using Live Update.:

```
bmide_modeltool.bat -u=username -p=password -g=dba -tool=all
                    -mode=upgrade -target_dir="TC_DATA"
```

bmide_remove_template

Validates removal of a template, cleans up instances of supported data model elements, and removes the template.

SYNTAX

```
bmide_remove_template -u=user-id {-p=password | -pf=password-file} -g=group
-mode=[remove | dryrun | instanceDelete]
-template=template
[-log=log-file-path]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies one of the following operation modes:

- **remove**

Looks for instances in database and if not found, then proceeds with removal of the template.

- **dryrun**

Reports all instances of the data model in given template including referencers of each instance.

- **instanceDelete**

Exports all of the instances into a TcXML file and then deletes them from the database.

-template

Specifies the name of the template to be removed.

-log

Specifies the full path and filename for the destination log file of utility execution results. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

bmide_setupknowledgebase

Generates the CLIPS (C Language Integrated Production System) rule file and uploads it to the database. It should only be run after upgrading the database from a pre-Teamcenter 2007 version to Teamcenter 2007 or later version.

SYNTAX

```
bmide_setupknowledgebase [-u=user-id {-p=password | -pf=password-file} -g=group]  
-regen=true/false -log=output-file-for-the-log-file [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-regen

Regenerates the CLIPS file. Values can be **true** or **false**.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

business_model_extractor

Extracts the data model definitions in the database into a XML file.

SYNTAX

```
business_model_extractor  
[-u=user-id {-p=password | -pf=password-file} -g=group]  
-outfile=output-file  
-mode=[all | schema | localization]  
-log=log-file  
-stats  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-outfile

Specifies file path and name of the XML file to be created with business data.

-mode

Specifies the extraction mode:

- **all**

Extracts the entire model. This is the default mode.

- **schema**

Extracts the classes and attributes.

- **localization**

Extracts all localizations in the localization's format.

-log

Specifies file path and name of the log file that contains the results of this execution.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To extract the entire datamodel in the database and output to the **db_extract.xml** file:

```
business_model_extractor -u=admin-user -p=admin-password -g=dba
-mode=all -outfile=c:\temp\db_extract.xml
```

business_model_updater

Deploys **tcschema**, types, and business rules.

Note:

Database statistics must be updated following any **business_model_updater** actions that cause indexes to be created or updated. Without updated statistics, the indexes are not utilized correctly by the database. For example:

```
exec dbms_stats.gather_schema_stats
(ownname=>'TCDBA',estimate_percent=>100,method_opt=>
'FOR ALL COLUMNS SIZE 1',degree=>8,cascade=>
true,no_invalidate=>FALSE);
```

For information about updating database statistics, see *Using database statistics* in *Teamcenter Site Consolidation*.

SYNTAX

```
business_model_updater [-u=user-id {-p=password | -pf=password-file} -g=group]
-mode= [install | upgrade]
-update= [ all | non_schema | schema | main_types | types | rules |
  bmf_rules_skip_missing_types | bmf_rules | lovs | lov_attaches |
  non_schema_ignore_lov_attach | convert_to_primary]
-process= [add | delete | change | all]
-file=XML-file-path-name
-log=log-file-path-name
[-mergedatasettype] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies the update method:

install

Specifies the update is for an installation of a new database.

upgrade

Specifies the update is for upgrading from one version to another version.

-update

Specifies which Business Modeler IDE object to update.

If you specify **non_schema_ignore_lov_attach**, the utility skips **TcLOVAttach** objects and processes all other **non_schema** objects.

-process

Specifies which Business Modeler IDE process to update.

-file

Specifies the path name to the XML file generated by Business Modeler IDE. The file contains Business Modeler IDE object definition data such as **Class**, **Type**, and **LOV** attachments.

-log

Specifies the path name to the log file.

-mergedatasettype

Specifies that dataset type definitions must be merged with existing dataset type definitions.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

change_datasets

Modifies the dataset reference name and the associated tool. This utility is run when resolving dataset type name collisions.

The utility generates the same package that you can generate from within Business Modeler IDE. See *Type name collision error in BMIDE for Data Model Design*.

SYNTAX

```
change_datasets [-u=user-id {-p=password | -pf=password-file} -g=group]
-ds_info=change-dataset-information-xml-file-location
-keep_definition
-dry
-bulk
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ds_info

Specifies the full path to the **ChangeDatasetInfo.xml** file generated by the utility.

-keep_definition

Specifies to keep the original tool and reference definition.

-dry

Logs information about modifications. No actual modifications are done with this argument. This is an optional argument.

-bulk

Runs a bulk update. This is an optional argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

change_type_name

Renames a custom business object type in the database. When two business object types have the same name, a collision occurs in the database. Use this utility to change the name of one of the types so that there is no longer a collision.

Warning:

Before renaming business objects, read *Rename a business object* in *BMIDE for Data Model Design*. You can damage your database if you do not perform all the manual steps mentioned in the documentation.

SYNTAX

```
change_type_name [-u=user-id {-p=password | -pf=password-file} -g=group]
[-old_type=current-name-of-business-object
-new_type=new-name-of-business-object]
-config=configuration-file
-dry
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. This argument is mutually exclusive with the **-pf** argument.

If neither this argument nor the **pf** argument is used, the system assumes the *user-ID* value to be the password.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

If neither this argument nor the **pf** argument is used, the system assumes the *user-ID* value to be the password.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-old_type

Identifies the business object to be renamed.

-new_type

Specifies the new name to replace the existing business object name.

-config

Optional. Identifies a configuration XML file that contains processing instructions. If this argument is not used, the default *TC_DATA\change_type_name_config.xml* configuration file is used.

-dry

Optional. Logs information about modifications. No actual modifications are done with this switch.

-h

Displays help for this utility.

EXAMPLE

To run the utility in dry run mode, enter the following command on a single line:

```
change_type_name -u=username -p=password -g=dba
-old_type=oldItem -new_type=newItem -dry
```

This runs the utility in dry run mode to estimate how much data would be altered if the old business object is renamed.

clips_dataset_upload

Stores both the ASCII (text) version and binary version of the CLIPS knowledge base files into the CLIPS (singleton) dataset in Teamcenter. This utility is called during Business Modeler IDE deployment activities when changes are identified to condition objects or rules based framework objects (application extension point and application extension rules). Only administrative users are allowed to run this utility.

SYNTAX

```
clips_dataset_upload [-u=user-id {-p=password | -pf=password-file} -g=group]
-file=path-to-upload-file
-log=output-file-for-the-log-file
-force_uncheckout=true/false
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the full path to the file specified for upload. Read access must be allowed on the directory.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-force_uncheckout

Cancel checkout of the CLIPS rules dataset and continue processing if set to **true**. If set to **false**, return an error if the CLIPS rules dataset is checked out. Values can be **true** or **false**.

This is typically used to restore the CLIPS rules dataset if left checked out in error.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

convert_forms

Allows a user with DBA privileges to convert legacy file-based forms to storage-based forms. You can manage the conversion process, as follows:

1. Determine whether a given form type or all form types to be converted.
2. Define attribute mapping between the file-based and storage-based forms.
3. Control the conversion process by specifying the number of forms to be converted during each run of the utility. Before performing the conversion, you can run the utility to generate an output file containing the UIDs of the file-based forms. This allows you to formulate a plan for performing the conversion by distributing the workload between multiple runs on multiple machines, if necessary.
4. Run the utility in batch mode without user intervention.
5. Restart the process without data corruption in the event that the process is stopped or terminates abnormally.
6. Generate log files listing information about forms that were successfully converted, failed to convert, and which attribute values are dropped or truncated. These files are retained if the process is terminated before completion.

UPGRADING FORMS

Updating file-based forms to storage-based forms involves the following steps:

1. Generate a file containing the list of forms to be converted, by running the **convert_forms** utility, as follows:

```
convert_forms  
-identify -output_file=file-name [-type=type-name]
```

This produces an output file containing the UIDs of file-based forms, one per line. The output file must be opened in **append** mode. This allows multiple lists of form type information to be contained in the same file. If the **-type** argument is not specified, all file-based forms are included in the output file.

2. Run the utility, as required, from one or multiple machines, as follows:

```
convert_forms  
-convert -process_file=file-name [-input_options=file-name]
```

3. If errors occur during the conversion process, the UIDs of the forms that were not converted are listed in the error file. After identifying and correcting the errors, you can use the **ErrorFile** file as the input file when rerunning the utility to convert the forms.

The **-process_file** argument specifies a file containing information that could be specific to each job. The **-input_options** argument specifies a file that is common to all jobs.

PROCESS OPTIONS FILE

The **-process_file** argument specifies a file containing information related to specific runs of the utility. You can copy the following example, paste it into a text editor, and use it as a starting point for your process options file:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<ProcessInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="convertFormProcessInfo.xsd">
  <InputFile>C:\\temp\\form_uids.txt</InputFile>
  <StartLine>1</StartLine>
  <EndLine>100000</EndLine>
  <LogFile>C:\\temp\\log.txt</LogFile>
  <ErrorFile>C:\\temp\\error.txt</ErrorFile>
  <SuccessFile>C:\\temp\\success.txt</SuccessFile>
</ProcessInfo>
```

InputFile	Specifies either the file generated as output when the utility is run with the -identify option or the ErrorFile file generated during a conversion run.
StartLine	The line number in the InputFile that specifies the beginning of the block of forms to be converted. If the line number is not specified, the default value is 1 .
EndLine	The line number in the InputFile file that specifies the end of the block of forms to be converted. If the line number is not specified, the default end line is the end of the file.
LogFile	Specifies the name of the file that logs information about dropped or truncated attribute values.
ErrorFile	Specifies the name of the file containing the UIDs of forms that were not converted due to errors. Errors encountered during conversion do not stop the process. When the reasons for the failure have been identified and corrected, the ErrorFile file can be used as the input file when the utility is rerun to convert those forms.
SuccessFile	Contains UIDs of forms that were successfully converted. This file can be useful for multi-site conversions.

All three files, **LogFile**, **ErrorFile**, and **SuccessFile**, are optional. If not specified, the corresponding file is not generated.

INPUT OPTIONS FILE

Unlike the process options file, which is specific to a particular run of the utility, the file specified by the **-input_options** argument contains information that is common to all runs of the utility. The following example illustrates the format of an input options file:

```
<FormTypes>
  <Type name=<name>>
    <DropAttrs action=none | all | unmapped | DropList>
      <ImanFileAttr name=<name> log=no | yes />
      .....
    </DropAttrs>
    <MapAttrs>
      <Map ImanFileAttr=<name> POMAttr=<name> truncate=no | yes |
log />
      .....
    </MapAttrs>
    <KeepLastModified action=no | yes />
    <DeleteImanFile action=yes | no />
  </Type>
  .....
</FormTypes>
```

All element attributes in the file have default values, indicated in *italics* in the example. If all default values are assumed, the input options file can be omitted.

Type	One or more Type elements can exist in the mapping file. Forms to be converted can be of different form types, all of which are listed in the file. If a form has no corresponding Type element, all form ImanFile attributes are mapped to the corresponding POM storage class attributes with the same names.
DropAttrs	Each DropAttrs element can contain zero or more DropList elements. DropAttrs action=none indicates that no attributes are dropped, all indicates that all attributes are dropped and no storage object needs to be created. unmapped indicates that all unmapped attributes are dropped. DropList indicates that a list is created of attributes that are dropped.
ImanFileAttr	To use the ImanFileAttr element, the DropList action must be used for the DropAttrs element. The ImanFileAttr element has a log attribute. If the value of this attribute is yes, the dropped attribute name and value are written to the log file.
MapAttrs	Each MapAttrs element can contain one or more Map elements. The Map element allows an ImanFile attribute to be mapped to a POM storage class

attribute with a different name. If an **ImanFile** attribute is not included in this list, it is mapped to the storage class attribute with the same name.

The **Map** element allows you to truncate the string data on conversion. If the **truncate** attribute value is **no** an error occurs and the form is not converted. If the value of the **truncate** attribute is **yes**, the data is truncated and not logged on the log file, while the **log** attribute will truncate the data and log it in the log file.

Only primitive attribute types are supported by this utility. Date, typed, and untyped references are not supported.

Do not create empty storage objects. Create an object only if one or more values are copied from the **ImanFile** properties.

Form attributes are case sensitive.

KeepLastModified	The KeepLastModified element specifies whether the last modified date must be updated to reflect the time of conversion. The default value is to update the last-modified date.
DeleteImanFile	The DeleteImanFile element specifies whether to delete the ImanFile after conversion. The default value is to delete the file.

You can copy the following example, paste it into a text editor, and use it as a starting point for your input options file:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<FormTypes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="convertFormInputOptions.xsd">
  <Type name="UGPartAttr">
    <DropAttrs action="DropList">
      <ImanFileAttr name="ASSEMB_NO" log="yes" />
      <ImanFileAttr name="MODEL" log="yes" />
    </DropAttrs>
    <MapAttrs>
      <Map ImanFileAttr="CREATOR" POMAttr="CREATOR" truncate="log" />
    </MapAttrs>
    <KeepLastModified action="no" />
    <DeleteImanFile action="no" />
  </Type>
</FormTypes>
```

SYNTAX

```
convert_forms [-u=user-id {-p=password | -pf=password-file} -g=group]
-identify -output_file=file-name [-type=form-type] -convert -process_file=file-name [-
input_options=file-name]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-identify

Generates an output file containing the UIDs of file-based forms. This argument is used in conjunction with the **-output_file** argument.

-output_file

Specifies the name of the output file.

-type

Specifies the type of the forms to be converted. If not specified, all file-based forms are converted.

-convert

Converts file-based forms that are read from a previous run of the utility using the **-identify** option. The **-convert** argument is used with the **-process_file** and **-input_options** arguments.

-process_file

Specifies the process file containing names of the input file, log file, error file, and success file, as well as the start line number and end line number. For more information about this file, see *Process options file*.

-input_options

Specifies the name of an input file containing information about how to convert forms. For more information about this file, see *Input options file*. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

The **convertFormProcessInfo.xsd** and **convertFormInputOptions.xsd** XML schema files are delivered as part of your Teamcenter installation and are located in the **imandata** directory. You must use these schema files to process the XML files that you generate.

RESTRICTIONS

None.

EXAMPLES

- To output the UIDs of all file-based forms of **UGPartAttr** type into the **C:\temp\form_uids.txt** file, enter the following command on a single line:

```
convert_forms -u=admin-user -p=admin-password -g=dba -identify
               -output_file=C:\temp\form_uids.txt -type=UGPartAttr
```

The **C:\temp\form_uids.txt** file is used in the example in *Process options file*.

- To convert file-based forms, reading in the **C:\temp\process_info.txt** file containing process information and the **C:\temp\options_file.txt** containing conversion information, enter the following command on a single line:

```
convert_forms -u=admin-user -p=admin-password -g=dba -convert
               -process_file=C:\temp\process_info.txt
               -input_options=C:\temp\input_options.txt
```


deploy_archive

Archives deployment files for both Teamcenter Environment Manager (TEM) and the Business Modeler IDE. An administrator can run this utility from the command line in **retrieve** mode to retrieve the last set of deployment archive files.

This utility is automatically run at deployment to archive files, but you can turn off this functionality by setting the **DEPLOY_ARCHIVE_ACTIVE** environment variable to **false** (or **off**, **no**, or **0**).

The deployment archival process is as follows:

1. A set of extractor and updater utility log files (specified by an input directory) are zipped into an archive file.
2. A set of compare history files associated to the specific deployment are zipped into an archive file.
3. For every deployment, the zipped archive files are uploaded into a new deploy archive dataset (**Fnd0BMIDEDeployArchive** dataset type).

The deployment archive file retrieval process is as follows:

1. Locate the archive datasets (**Fnd0BMIDEDeployArchive** dataset type) from the Teamcenter database using the **BMIDEDeployArchiveRecovery** saved query.
2. A subdirectory using the dataset name with the create date/time stamp is created under a specified target directory.
3. Each named reference (archive zip file) for the archive dataset is exported to that subdirectory.

For more information, see *Retrieve deployment archive files* in *BMIDE for Data Model Design*.

SYNTAX

```
deploy_archive -u=user-id {-p=password | -pf=password-file} [-g=group]
-action=[deploy | retrieve]
-dir=source-or-target-file-path
-deploySource=[tem | bmide]
-retrieveNum=number-of-archives
-afterDate=date-and-time
-log=log-file-path
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

Creates or retrieves the archive. You must specify one of these options:

- **deploy**

Collects and creates the deploy archive ZIP files and upload them into a deploy archive dataset. Requires the **-deploySource** argument to be specified.

- **retrieve**

Extracts the deploy archive ZIP files from the specified deploy archive datasets. The dataset name is used to create a subdirectory in the OS under the directory name given by the **-dir** parameter. The named references (archive ZIP files) are exported to that subdirectory.

-dir

Specifies the full path to the source directory (when using the **-action=deploy** argument) or to the target directory (when using the **-action=retrieve** argument). The source directory must exist and have read access and contains files to archive for the deploy action. The target directory must exist and have write access and contains subdirectories (by dataset name) that each contain ZIP archive files from the retrieve action.

-deploySource

Identifies the source of the deployment activity when using the **-action=deploy** argument. (Ignored when the action is **retrieve**.) The dataset name format is **deploy_tem** | **bmide_user-id_date-time-stamp**. Valid values are:

- **tem**

Specifies that Teamcenter Environment Manager is the source of the deployment.

- **bmide**

Specifies that the Business Modeler IDE is the source of the deployment.

-retrieveNum

Qualifies the number of archive datasets (from latest to earliest) to return from the saved query execution when the **-action=retrieve** argument is used. (Ignored when the action is **deploy**.) The default is **3**. If not specified, 3 datasets are retrieved. If **0** is set, then all datasets are retrieved. The actual number of datasets exported can be different than this value based on the total number of deploy archive datasets.

-afterDate

Specifies the time after which to collect archive log files when using the **-action=deploy** argument. (Ignored when the action is **retrieve**.) The required format is *yyyy-mm-dd hh:mm:ss*.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

execute_rbf_rules

Executes the specified application extension rule. The utility validates the application extension rules are deployed and functioning correctly. Input parameters for the rule execution are specified in a file. Output is written to a specified file or displayed on the console. Execution details are written to a specified log file or displayed on the console.

DESCRIPTION

SYNTAX

```
execute_rbf_rules -u=user-id {-p=password | -pf=password-file} [-g=group]
-id=application-extension-point-id -inputfile=input-file-name
[-outputfile=output-file-name] [-log=log-file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-id

Specifies the ID of the application extension point.

-inputfile

Specifies the location of the input file. The input file format is:

```
Column Name | Datatype | Value
```

There is one line for each input parameter.

Datatype is one of the following:

```
String
Integer
Double
Float
Boolean
Date
Tag
```

See *Examples* for an example of the input file.

-outputfile

Specifies the location of the output file. The output file format is the same as the input file format. Datatype for a output parameter is one of the following:

```
String
Integer
Double
Float
Boolean
Date
```

-log

Specifies the location of the log file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following is an example of the utility:

```
execute_rbf_rules -u=admin-user -p=admin-password -g=dba  
-id=tc.core.pse.icon -inputfile="d:\in1.txt"  
-outputfile="d:\out1.txt" -log="d:\rbf1.log"
```

- The following input file example specifies that there are two input columns, **Material** and **Pressure**, with corresponding datatypes of **String** and **Integer**. The value of **Material** is **Steel**; the value of **Pressure** is **10**.

Material	String	Steel
Pressure	Integer	10

getglobalconstantvalue

Returns the value of a particular global constant in the database. The utility provides help in troubleshooting issues on the server once the global constants are deployed. The utility accepts the name of a global constant and outputs the value of the constant, if present.

SYNTAX

```
getglobalconstantvalue [-u=user-id {-p=password | -pf=password-file} -g=group]
-constantname=global-constant-name [-v= on | off] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-constantname

Specifies the name of the global constant.

-v

Specifies verbose mode. Specify **on** to display a detailed message. Specify **off** to display only the value. Default value is **on**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Entering the following command:

```
getglobalconstantvalue -u=admin-user -p=admin-password -g=dba  
-constantname=SampleConstant
```

returns the following:

```
Value of the global constant SampleConstant = XYZ
```


get_key_definition

Gets the multifield key definition for a business object type. *Multifield keys* are identifiers assigned to each object to ensure their uniqueness in the database.

At a command prompt, type:

```
get_key_definition -class=business-object-type
```

The results are displayed as:

Key Definition for *business-object-type* is *multifieldkey-definition*

Note:

There are other utilities that you can use to work with keys. Use the **get_key_string** utility to get the multifield key value of an item instance as a string containing property names and values, and use the **mfk_update** utility to update multifield key definitions in the key table in the database.

For more information, see *Analyzing multifield keys* in *BMIDE for Data Model Design*.

SYNTAX

```
get_key_definition
-class=business-object-type
[-h]
```

ARGUMENTS

-class

The name of the business object type for which you want to get the multifield key definition.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To get the multifold key definition for a business object type named **T5_MyItem**:

```
get_key_definition -class=T5_MyItem
```

The results are displayed as follows:

```
Key Definition for T5_MyItem is T5_MyItem{item_id,object_type}
```

get_key_string

Gets the multifield key value of an item instance as a string containing property names and values. This utility can be run on the **Item** business object or any of its children. *Multifield keys* are identifiers assigned to each object to ensure their uniqueness in the database.

If the **-item** or **-key** argument input corresponds to more than one item, the key values for each of the items found is displayed. If the **-key** argument is used, partial keys are allowed. For example, if the key for an item is defined as **item_id,prop1,prop2**, inputs such as **-key=item_id=123456,prop1=abcd** are allowed, which may result in multiple items being found and displayed.

To use the **-item** argument to get key values for an item, use the following command:

```
get_key_string -item=item-ID
```

The results are displayed as:

```
Item Type is business-object-name, Key String is property1=property1-
value, property2=property2-value, etc
```

If other properties are added to the multifield key definition for an item business object, they are displayed separated by commas. For example, if the **item_id** and **object_type** properties comprise the multifield key for a custom item business object named **T5_MyItem**, the output of the command is:

```
Item Type is T5_MyItem, Key String is item_id=000016,
object_type=T5_MyItem
```

To use the **-key** argument to get the business object type for an item, use the following command:

```
get_key_string -key=property=property-value
```

The results are displayed as:

```
Item Type is business-object-name,
Key String is property=property-value
```

Note:

- There are other utilities that you can use to work with keys. Use the **get_key_definition** utility to get the key definition for a business object type, and use the **mfk_update** utility to update multifield key definitions in the key table in the database.
- To change the delimiters used in the input for the **get_key_string** utility, create the **TC_KEY_STRING_DELIMITER_VALUES** preference.

For more information, see the *Teamcenter Environment Variables*.

For more information, see *Analyzing multifield keys* in *BMIDE for Data Model Design*.

SYNTAX

```
get_key_string
-item=business-object-item-ID
-key=key-string
[-h]
```

ARGUMENTS

-item

Specifies the ID of items for which key values are to be displayed.

-key

Specifies the key value of items for which item types are to be displayed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To get the values of the multifield key properties on an **Item** business object instance whose item ID is **000016**:

```
get_key_string -item=000016
```

The results are displayed as follows:

```
Item Type is Item, Key String is item_id=000016,object_type=Item
```

- To get the values of the multifield key properties on an instance of a custom item business object (**T5_MyItem**) whose item ID is **000019**:

```
get_key_string -item=000019
```

The results are displayed as follows:

```
Item Type is T5_MyItem,  
Key String is item_id=000019,object_type=T5_MyItem
```

- To get the item type of an instance whose key is **item_id=000016**:

```
get_key_string -key=item_id=000016
```

The results are displayed as follows:

```
Item Type is Item, Key String is item_id=000016
```

getpropertyconstantvalue

Returns the value of a property constant on a particular business object and property in the database. The utility provides help in troubleshooting issues on the server once the property constants are deployed to the database. The utility also applies the inheritance and scope of the property constants while fetching the value. Therefore, it removes the burden of manually analyzing the extracted model to troubleshoot a property constants value on a given business object and property. The utility accepts the name of a property constant, business object and property and outputs the value of the constant, if present.

SYNTAX

```
getpropertyconstantvalue [-u=user-id {-p=password | -pf=password-file} -g=group]
-constantname=constant-property-name -typename=type-object-name
-propertyname=property-name [-v= on | off] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-constantname

Specifies the constant property name.

-typename

Specifies the name of the business object.

-propertyname

Specifies the name of a property on the business object specified by the **-typename** argument.

-v

Specifies verbose mode. Specify **on** to display a detailed message. Specify **off** to display only the value. Default value is **on**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Entering the following command:

```
propertyconstantvalue -u=admin-user -p=admin-password -g=dba
    -constantname=Visible -typename=Item -property=item_id
```

returns the following:

```
Value of the property constant Visible on Item.item_id = true
```

gettypeconstantvalue

Returns the value of type constant, that is, a business object constant, on a particular business object in the database. The utility provides help in troubleshooting issues on the server once the type constants are deployed to the database. The utility also applies the inheritance and scope of the type constant while fetching the value. Therefore, it removes the burden of manually analyzing the extracted model to troubleshoot a type constants value on a given business object. The utility accepts the name of a type constant and business object and outputs the value of the constant, if present.

SYNTAX

```
gettypeconstantvalue [-u=user-id {-p=password | -pf=password-file} -g=group]
-constantname=type-constant-name -typename=type-name [-v= on | off] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-constantname

Specifies the name of the type constant.

-typename

Specifies the name of the business object.

-v

Specifies verbose mode. Specify **on** to display a detailed message. Specify **off** to display only the value. Default value is **on**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Entering the following command:

```
gettypeconstantvalue -u=admin-user -p=admin-password -g=dba
                    -constantname=SampleConstant -typename=WorkspaceObject
```

returns the following:

```
Value of the type constant SampleConstant on type
WorkspaceObject = QWERTY
```

manage_icon_files

Manages uploading icon files to the database. *Icons* are images placed on business object instances to identify them in the user interface.

When templates are deployed to a server from the Business Modeler IDE or installed using Teamcenter Environment Manager, the template's zipped icon files are placed into the **TC_DATA/model/icons (TC_DATA_MODEL)** directory. At deployment or installation time, the **manage_icon_files** utility is used internally by Teamcenter to upload the icons into **FndOIconResource** dataset instances in the database. To search for these datasets using the rich client, search for the **Icon Resource Dataset** type.

For more information, see *Add or change a business object icon* in *BMIDE for Data Model Design*.

SYNTAX

```
manage_icon_files -u=user-id {-p=password | -pf=password-file} [-g=group]
-option=upload
-template=template-name
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option=upload

Uploads all icon files from the *TC_DATA_MODEL/icons* directory to datasets for the templates specified in the **-template** argument.

-template

Specifies the templates to upload. Multiple template are specified in a comma-separated list, for example, **foundation,tcae**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
manage_icon_files -u=admin-user -p=admin-password -g=dba
-option=upload -template=bmideproject
```

manage_model_files

Manages upload and download of files in the *TC_DATA_MODEL* directory into **Fnd0BMIDEResource** datasets. These Business Modeler IDE datasets are stored in the database in a **Fnd0BMIDEResource** folder. You can query for this folder in the rich client for quick reference to the Business Modeler IDE resource files that are maintained in the database. You can use these files to restore your Business Modeler IDE environment.

For more information, see *Back up project data in BMIDE for Data Model Design*.

SYNTAX

```
manage_model_files -u=user-id {-p=password | -pf=password-file} [-g=group]
-option=[list | upload | download] [-dir=directory -template=template
-release=Teamcenter-release -resource=project ]
-syncToDb
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option

Valid values for option are:

- **list**

Lists the version of all templates, language, and model files in the utility output by querying the versions from the datasets.

- **upload**

Uploads all template-related files from the *TC_DATA_MODEL* directory to datasets. If **-template** is specified, files specific to the given template are uploaded from the *TC_DATA_MODEL* directory to datasets.

- **download**

Downloads all template-related files from datasets to the *TC_DATA_MODEL* directory. If **-template** is specified, files specific to the given template are downloaded from datasets to the *TC_DATA_MODEL* directory. If **-dir** is specified, files are downloaded to the given directory instead of the *TC_DATA_MODEL* directory.

-dir

Absolute directory path where templates should be downloaded. This can be used only with the **-option=download** option.

-template

List of comma-separated template file names, for example, **foundation,tcae**.

-release

Teamcenter version, for example, **tc9000.1.0**. If the **-release** argument is specified, the template project backups for that release are retrieved. This argument can be used with only the **-resource=project** argument.

-resource=project

Downloads the template project backup to the directory specified by the **-dir** argument. The **-dir**, **-release**, and **-template** arguments must be used in conjunction with the **-resource=project** argument.

-syncToDb

Runs the extractor and uploads the extracted model file into the database. Run the utility with this option only if the data in the database is modified and the model file in the database is not synchronized.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To retrieve the latest backup of the **customer** template project to the local **D:\BackupDownloadDir** directory:

```
manage_model_files -u=username -p=password -g=dba  
-option=download -template=customer -dir=D:\BackupDownloadDir  
-resource=project -release=tc9000.1.0
```

mfk_update

Updates multifield key definitions in the key table in the database. Normally, during upgrade the system runs this utility so that business object instances on the server are migrated to the new multifield key definitions. As an administrator, you can also use this utility to:

- Evaluate proposed multifield key definitions in a template before installing the template to the production server. Specifically, you can determine whether a multifield key is deployable or not (that is, unique across business object instances in the particular domain, all properties found and of the right type, and so on).

The utility reports potential collisions (duplicate multifield key values) and the quantity of collision instances.

- Analyze the multifield key definitions on the server, and if there are corrupt or inconsistent key definitions, rebuild the key table on the database.
- Rebuild all multifield key values in the system, thereby guaranteeing correctness.

Note:

There are other utilities that you can use to work with keys. Use the **get_key_definition** utility to get the multifield key definition for a business object type, and use the **get_key_string** utility to get the multifield key value of an item instance as a string containing property names and values.

For more information, see *Analyzing multifield keys* in *BMIDE for Data Model Design*.

SYNTAX

```
mfk_update -u=user-id {-p=password | -pf=password-file} [-g=group]
-check
-file=template-file-path
-rebuild
[-log=log-file-path] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter

database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-check

Runs the utility in analysis mode. When analyzing proposed multifield key definitions, the **-file** option must specify the file name.

-file

Specifies the path of the XML file that contains multifield key definitions to be analyzed. If **-file** option is not specified, the utility analyzes the multifield key definitions in the system.

-rebuild

Rebuilds all multifield key entries. This argument is mutually exclusive with the **-check** argument.

-log

Specifies the path of the log file that contains results of the multifield key update. The default is the `TC_TMP_DIR/mfk_updater_date-and-time.log` file. If the `TC_TMP_DIR` environment variable is not set, the log file is created in the system temporary directory (`C:\TEMP` on Windows or `/tmp` on Linux).

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To perform an analysis of proposed keys in a custom Business Modeler IDE template:

```
mfk_update -u=admin-user -p=admin-password -g=dba -check  
-file=C:\delta.xml  
-log=C:\mfk_check_template.log
```

- To perform an analysis of the keys in the database:

```
mfk_update -u=admin-user -p=admin-password -g=dba -check  
-log=C:\mfk_check_database.log
```

- To rebuild the key table for all multifield key definitions in the database:

```
mfk_update -u=admin-user -p=admin-password -g=dba -rebuild  
-log=C:\mfk_rebuild.log
```

migrate_propagation_data

Details about the background of this utility and when to use it are described in **SFB-Teamcenter-8001528**. The bulletin title is *ProjectObjectRelation (POR) proliferation*.

Issue: For objects created prior to Tc 11.2.2 that have security attributes assigned, the following undesired data conditions can exist after upgrading to 11.2.2 or 11.2.3:

- Unnecessary ProjectObjectRelation (POR) objects are created during various actions which trigger propagation.

Affected Software:

Teamcenter 11.2.2 and 11.2.3, and environments upgraded to Teamcenter 11.2.2 or 11.2.3 that use:

- Project level security
- ADA Licenses
- IP_Classification Gov_Classification and security attribute propagation Project Object Relation (POR) proliferation

The underlying issue is resolved in the following Teamcenter releases and later releases in the patch and minor release series. These release streams contain the server code changes to address the proliferation issue.

- Tc11.2.2.1_a01_4
- Tc11.2.2.1_b01_4
- Tc 11.2.3.1_a01_1
- Tc 11.2.3_c01_1
- Tc 11.3

Impact:

- Proliferation of superfluous POR objects

- Performance and lock issues with objects having legacy PORs when the data with security attribute assigned is used
- Propagation functionality is unaffected

Remedy:

Apply the "one time use"
migrate_propagation_data utility.

Fixes and migrates legacy **ProjectObjectRelation** (POR) propagation data in order to support multiple propagation groups.

This utility can remedy the following data conditions:

- Duplicate propagation groups for the same security attributes
- Data conditions with "No Group" redundant values
- Empty propagation group values

All these data conditions are analyzed in scope of common owning top-level object.

SYNTAX

```
migrate_propagation_data [-u=user-ID {-p=password | -pf=password-file} -g=group]
[-mode=dryRunMode | fixMode]
[-migrate=1]
[-deleteSuperfluous=1]
[-deletePORWithAllNullAttr=1]
[-toplevelCleanupObjOutputFile=filename]
[-toplevelMigrationObjOutputFile=filename]
[-filePath=file path]
[-noGroupCount=1]
[-vlaCheckRequired=1]
[-defer_bp_inserts=1]
[-insert_deferred_bps=1]
[-fixSingleGroup=group]
[-bypass=1]
[-verbose=1]
[-superfluous_use_lsd=1]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the clear text password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-mode

Different modes for running the utility. Report is written to **syslog**. Set **TC_KEEP_SYSTEM_LOG** to true to keep the **syslog** file.

dryRunMode

In dry run mode, no modification is made to the data. The utility will report in the **syslog** the list of objects for which duplicate **ProjectObjectRelation** (POR) exist and the number of duplicate POR objects.

fixMode

In fix mode, duplicate **ProjectObjectRelation** (POR) associated with the objects are deleted. The utility will report the objects for which PORs are deleted in the **syslog** file.

-filePath

An input file containing UUIDs of objects which have PORs in need of processing; these are associated with the POR via top-level references. Specify each UUID on a separate line.

-migrate=1

Performs migration of objects provided in the owning top-level object UUID (objects having their own PORs) list in input file specified by the **-filePath** argument.

-deleteSuperfluous=1

Superfluous POR are POR where another POR for same owning top level object exists with same group association (for example, explicit group or "No Group" for the same security attributes).

Performs cleanup of superfluous POR in the database (no input file of UUIDs required) as follows:

- Propagation works correctly only when the object has PORs with valid propagation group values. Because PORs with propagation group as empty (NULL entry or empty string in the database column for propagation group) is not valid, propagation does not work as expected.

Clean all POR with ""/empty group and convert to "No Group"; this data condition may occur for Microsoft SQL Server platform only for releases $\geq 11.2.2$ and $< 11.2.3$. You can optionally specify a list of top-level object UUIDs.
- For POR with "No Group", find and remove any other POR associated to the same top-level objects. The security values on the superfluous PORs are merged with the corresponding security values on "No Group" POR.
- For POR with duplicate groups, find and remove all such duplicate PORs. The security values on the superfluous PORs are merged with the corresponding security values on same security group POR.

Note:

Both the **-migrate** and **-deleteSuperfluous** arguments can be run in **dryRunMode** as well (see usage below).

-deletePORWithAllNullAttr=1

Deletes POR having "No Group" and all null security attributes. These POR cannot be migrated and they are not required for re-propagation. However, note that this code will not scrutinize any additional custom security attributes. If such custom security attributes are used, it is advised not to invoke **fixMode** for this use case and leave these in the system.

-toplevelCleanupObjOutputFile

Runs a query against the entire database to find all UUID of top-level objects with "No Group" POR, which needs to be cleaned. You can use this file as the input for the **-filePath** option with the **-deleteSuperfluous** option to clean up such objects.

-toplevelMigrationObjOutputFile

Runs a query against the entire database to find all UID of top-level objects with “No Group” POR which need to be migrated. You can use this file as the input for the **-filePath** option with the **-migrate** option to migrate such objects.

-noGroupCount=1

Displays the total number of POR with “No Group” in the database

-vlaCheckRequired=1

Checks for **ProjectObjectRelation** (POR) with inconsistent project_list VLAs or license_list VLA (for example, VLA count different from actual number of projects/licenses).

-defer_bp_inserts

Defers inserting records to backpointer table. Until these deferred records are inserted back into the backpointer table, “where referenced” does not see POR objects referencing workspace objects for the newly cloned POR.

Customers with huge POR data which requires cloning can reduce POR migration time by deferring inserting records to backpointer table.

POR migration clones the PORs and some portion of time is spent in inserting records into backpointer table. These backpointer records maintain the references of cloned POR with top-level object.

The number of records to insert into the backpointer table is determined by the number of PORs, which need to be cloned during POR migration. The number of cloned PORs depends on security data conditions.

The deferred records are inserted to a temporary table that can be processed later using the **-insert_deferred_bps** option.

-insert_deferred_bps

Inserts deferred records for cloned POR from the temporary table into the backpointer table.

-fixSingleGroup=group

Performs migration/clone for a single group only when used with the migrate/clone mode.

-bypass=1

Allows the migration of objects even through duplicate PORs may exist and were not first cleaned when set as part of POR migration.

-verbose=1

Displays relevant debug information if the **verbose** mode is activated by default.

-superfluous_use_lsd=1

Applies to the superfluous POR deletion mode. When this argument is set, the winning POR from among a set of POR for the same top-level object and same group to be the one with latest lsd.

If not set, the POR with security attributes most closely matching those of the associated top-level object is chosen. This is the default behavior.

-h

Displays help for this utility.

EXAMPLES

Note:

The following is a sequence of usage commands:

- Find the number of "No Group" POR objects:

Usage 1.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -noGroupCount=1
```

- Cleanup command usage:

- Usage 2.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -toplevelCleanupObjOutputFile=filename
```

- Usage 3a.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=dryRunMode -deleteSuperfluous=1 -filePath=name
```

- Usage 3b.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -deleteSuperfluous=1 -filePath=name
```

- Usage 4a.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=dryRunMode -deleteSuperfluous=1
```

- Usage 4b.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -deleteSuperfluous=1
```

- Usage 4c.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -deleteSuperfluous=1 -verbose=1
```

- Usage 4c.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -deleteSuperfluous=1 -superfluous_use_lsd=1
```

- Migrate command usage:

- Usage 5.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        toplevelMigrationObjOutputFile=filename
```

- Usage 6a.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=dryRunMode -migrate=1 -filePath=name
```

- Usage 6b.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -migrate=1 -filePath=name
```

- Usage 7a.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -migrate=1 -filePath=name -defer_bp_inserts=1
```

- Usage 7b.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -insert_deferred_bps=1
```

- Usage 7c.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -migrate=1 -fixSingleGroup=group
```

- Usage 7d.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group
                        -mode=fixMode -migrate=1 -bypass=1
```


- Additional checks:

- Usage 8.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group  
-mode=fixMode -deletePORWithAllNullAttr=1
```

- Usage 9.

```
migrate_propagation_data -u=Tc-admin-user -p=password -g=group  
-vlaCheckRequired=1
```

package_live_updates

Packages live update templates stored in the database. Only templates that are enabled for live updates can be packaged using this utility.

For more information, see *Package live updates in the database* in *BMIDE for Data Model Design*.

SYNTAX

```
package_live_updates -u=user-id {-p=password | -pf=password-file} [-g=group]  
-template=template-name  
-dir=directory  
[-log=log-file-path] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-template

Specifies the live update template to package.

-dir

Specifies the full path to the directory where the template package is created. If the **-dir** argument is not provided, the package is created in the **TC_DATA_MODEL** directory. The name of the generated package template file is *template-name_template_extracted_from_live_update_site-name*.

-log

Specifies file path and name of the log file that contains the results of this execution. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

process_action_rules

Lists or deletes all the existing action rules in the database.

SYNTAX

```
process_action_rules [-u=user-id {-p=password | -pf=password-file} -g=group]
[-delete | -list] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-delete

Deletes all the existing action rules.

-list

Lists the actions rules configured for a site. This is the default mode if delete is not specified.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To list all action rules in the database:

```
process_action_rules -u=admin-user -p=admin-password -g=dba -list
```

- To delete all action rules in the database:

```
process_action_rules -u=admin-user -p=admin-password -g=dba -delete
```

propagation_doctor

This utility analyzes the propagation configurations to determine if any configuration issues exist in the database and generates a report. The report contains the configuration issues found and the remedy actions to fix those issues.

Syntax

```
propagation_doctor -u=user {-p=password | -pf=password-file} -g=group name -outputfile=output file path [-h]
```

Arguments

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-outputfile

Path of output file to report the list of propagation configuration issues found in the database.

-h

Displays help for this utility.

Note:

Whenever you find any propagation discrepancies or perform an upgrade, execute this utility to check if any configuration or data issue exists in your system.

This utility investigates the following configurations:

Configuration Issue: Problematic propagation loops that may cause a potential traversal risk.

Analyzes the propagation rules to determine if any of the propagation rules set have potential traversal risks. It reports the propagation rule sets that have potential to create performance problems affecting various aspects of system behavior, related to user interactions and other Create/Read/Update/Delete activities.

Note:

A user must deploy the propagation rules in the database before executing the utility.

Example: A user deploys the following propagation rules in the database:

```
<PropagationRule sourceType=" Dataset" relRefPropName="
IMAN_external_object_link" referenceType="Relation"
destinationType="ItemRevision" operation="All" traversalCondition="isTrue"
actionCondition="isTrue" propagationGroup="Security Group I"
direction="Forward" propagationStyle="Merge" background="false"
secured="false"/>
```

After deploying the propagation rule, the user runs the utility command:

```
propagation_doctor -u=userid -p=pwd g=grp -outputfile=<path of the
outputfile>
```

This rule is problematic therefore the utility will report the following information in the output file:

Non COTS propagation rules in combination with COTS propagation rules could cause a potential traversal risk for the propagation paths below which could be forming a loop.

A Loop is formed when traversal starts from a type and repeatedly encounter the same or sub/super type.

This could cause potential performance problems for propagation related use cases.

Inspect non-COTS propagation rules in each of the following problematic propagation loops and follow one of the solution proposals below in

order to fix those non-COTS rules:

1. Re-formulate the rule with more specific source/destination classes.
2. Navigate to the relevant destination objects from some other source objects.
3. Use custom post actions to populate the data instead of using a propagation rule.
4. Re-evaluate whether this particular propagation rule is required in the first place.

For more detail refer to the Teamcenter Documentation section Identify and fix propagation rules with potential traversal risk.

Loop 1 Forward AbsOccData ImanRelation Dataset

Forward Dataset IMAN_external_object_link ItemRevision

Forward WorkspaceObject IMAN_manifestation WorkspaceObject

Reverse PSBOMViewRevision qualifier_bvr AbsOccDataQualifier

Reverse AbsOccDataQualifier data_qualifier AbsOccData

Loop 2 Forward WorkspaceObject IMAN_specification WorkspaceObject

Forward Dataset IMAN_external_object_link ItemRevision

Loop 3 Forward WorkspaceObject IMAN_UG_altrep WorkspaceObject

Forward Dataset IMAN_external_object_link ItemRevision

Loop 4 Forward WorkspaceObject TC_Attaches WorkspaceObject

Forward Dataset IMAN_external_object_link ItemRevision

Note:

After running the utility, four problematic propagation loops are found. The user should fix this non-COTS propagation rule by referring to the guidelines provided in Identify and fix propagation rules with potential traversal risk.

Configuration Issue: Propagation rules using Operation Type other than "All".

Reports propagation rules with operation type other than "All" in the output file as below. Creation of propagation rules with values for operation type other than "All" is disabled in BMIDE.

Configuration Issue: Propagation rules using Operation Type other than "All".

The propagation rules defined with operation type other than "All" are supported but we strongly encourage you to convert these propagation rules to operation type "All" so that all the propagation rules are consistent.

Going forward, we will deprecate the propagation rules defined with operation type other than "All".

The following propagation rules with operation type other than "All" are found in the Teamcenter database.

For each propagation rule, check if an equivalent propagation rule with "All" operation type already exists,

if yes then delete the existing propagation rule.

If an equivalent propagation rule with "All" operation type does not exist,

then create an equivalent propagation rule with "All" operation type and then delete the existing propagation rule.

Revise Forward ItemRevision IMAN_Motion Dataset

Security Group II Merge

SaveAs Forward ItemRevision IMAN_Motion Dataset Security Group

III Merge

Configuration Issue: Discrepancies between style value in "TC_propagation_security_group_style" preference and propagation rules.

Analyzes the propagation rules and **TC_propagation_security_group_style** preference to determine if there are any discrepancies between the style value in the propagation rules and preference.

It also verifies whether all the propagation rules for a specific propagation security group are defined with the same style.

It reports the propagation rules where the style/group does not match with the preference value. Discrepancies between the style value in the propagation rules and preference can result in unexpected propagation behavior.

If the specific propagation group is defined with multiple styles in the propagation rules and the group/style exists in the **TC_propagation_security_group_style** the utility will report the following information in the output file:

```
If a Propagation Security Group/Style is defined in the preference
"TC_propagation_security_group_style" then the same Security Group/Style
should be used in propagation rules.
Security Group/Style defined in the preference take precedence over the
Group/Style used in Propagation Rules. Please ensure that the Security
Group/Style used in propagation rules are matching with Group/Style
defined in the preference. Also, all propagation rules for a specific
security group should have the same style.
Please ensure that the same style is used for a specific group in
propagation
rules. Group/Style discrepancies are found for propagation rules shown
below:
The preference dictates Order Style for Security Group III
But the following propagation rules are defined with Overwrite Style for
Security Group III
```

```
Overwrite Forward A5_9338051_Item IMAN_based_on A5_9338051_Item All
Overwrite Forward A5_9338051_ItemRevision IMAN_based_on
A5_9338051_ItemRevision All
```

If a specific propagation group is defined with multiple styles in the propagation rules and the group/style combination does not exist in the **TC_propagation_security_group_style**, the utility will report the following information in the output file:

```
Configuration Issue: Discrepancies of style value in
TC_propagation_security_group_style preference and propagation rules.
If a Propagation Security Group/Style is defined in the
preference TC_propagation_security_group_style
then the same Security Group/Style should be used in propagation rules.
Security Group/Style defined in the preference take precedence over
the Group/Style used in Propagation Rules.
Please ensure that the Security Group/Style used in propagation rules
are matching with Group/Style defined in the preference.
Also, all propagation rules for a specific security group should have
the same style.
Please ensure that the same style is used for a specific group in
propagation rules.
```

Group/Style discrepancies are found for propagation rules shown below:
No Style exists for Custom Group IX in the preference.
But following propagation rules are defined with Merge and Overwrite
Style for Custom Group IX
Merge Forward ItemRevision IMAN_Motion Dataset All
Overwrite Reverse Item items_tag ItemRevision All

Configuration Issue: Propagation rules using "Fnd0TraversalCondition" for Traversal Condition.

Fnd0TraversalCondition signature is not compatible with the new traversal engine starting with release 11.3. If this condition is used in propagation rules, then the propagation will not work as expected.

Reports propagation rules with **Fnd0TraversalCondition** condition in the output file as below.

Configuration Issue: Propagation rules using Fnd0TraversalCondition
for Traversal Condition.

The "Fnd0TraversalCondition" signature is not compatible with the new traversal engine used in release tc11.3 and up.
If this condition is used in propagation rules then propagation will not work
as expected. You are at release 'Teamcenter P12000.3.0.20190701.00',
either change the traversal condition from "Fnd0TraversalCondition"
to "isTrue" or use "Fnd0TraversalCondition2" for the propagation rules
shown below.

Forward Item bom_view_tags Match All Security Group II Fill

Configuration Issue: ProjectObjectRelation (POR) with "NULL" propagation group.

Reports PORs (ProjectObjectRelation) with "NULL" propagation security group in output file. Existence of such PORs in the database will impact the propagation behavior and result in Error Condition.

Configuration Issue: ProjectObjectRelation (POR) with "NULL" propagation
group.

POR with "NULL" propagation group should not exist in Teamcenter
Database, it is considered as Error Condition.
1 POR with Propagation Group as NULL found in Teamcenter database.
Please review the SFB-Teamcenter-8001528 and execute
migrate_propagation_data
utility with -deleteSuperfluous option to fix such POR.

Note:

The number of PORs listed above will be replaced depending on the number of such PORs existing in the database. For example, if you have 100 PORs in the database that number would then display 100.

Configuration Issue: 'No Group' ProjectObjectRelation objects (PORs).

Reports PORs with Propagation Group with value 'No Group'

Configuration Issue: 'No Group' ProjectObjectRelation objects (PORs).

Examine the Teamcenter database for the existence of PORs having Propagation Group as 'No Group'.

4 ProjectObjectRelation objects (PORs) with Propagation Group as 'No Group' found in Teamcenter database.
Please review the SFB-Teamcenter-8001528 to evaluate if it is required to run migrate_propagation_data utility.

Note:

The number of PORs listed above will be replaced depending on the number of such PORs existing in the database. For example, if you have 100 PORs in the database with "No Group" that number would then display 100.

Configuration Issue: Missing "No Group" propagation rules.

Reports propagation rules with propagation group other than 'No Group' for which there is no matching "No Group" propagation rule.

If the equivalent "No Group" propagation rules are not deployed in the system, the propagation for objects having 'No Group' PORs will not work as expected.

Configuration Issue: Missing "No Group" propagation rules.

If any PORs with "No Group" propagation security group exist in the database,
and you are deploying propagation rules for
Security Groups other than "No Group" then please ensure that matching
"No Group"
propagation rules are also deployed in the DB.
If those "No Group" propagation rules are not deployed then it will
impact the
propagation behavior for those "No Group" PORs
(they will not be processed).

PORs with the "No Group" propagation security group are found in the database and propagation rules with "No Group" are missing for below corresponding propagation rules. Please either deploy equivalent propagation rules for the "No Group" or if you are planning to migrate those "No Group" PORs using `migrate_propagation_data` utility then refer to the SFB-Teamcenter-8001528.

```
Forward ItemRevision IMAN_3D_snap_shot Dataset All isTrue isTrue
Forward ItemRevision IMAN_UG_wave_position Dataset All isTrue isTrue
```

Configuration Issue: 'No Group' ProjectObjectRelation objects (PORs) with null security attributes.

PORs with "No Group" and all security attributes as NULL is a bad condition. The utility reports PORs having propagation group as 'No Group' and empty security attributes in output file as below.

```
1 ProjectObjectRelation object (POR) with Propagation Group
as 'No Group' and null security attributes found in Teamcenter database.
Please review the SFB-Teamcenter-8001528 to evaluate if it is required
to run migrate_propagation_data utility.
```

Note:

The number of PORs listed above will be replaced depending on the number of such PORs existing in the database. For example, if you have 100 PORs in the database with "No Group" and empty security attributes that number would then display 100.

Configuration Issue: Superfluous ProjectObjectRelation objects (PORs).

In below scenarios, PORs are considered as superfluous:

- If a top level object has more than one POR, and one of the PORs is a "No Group" POR
- If a top level object has multiple PORs for the same propagation security group

Reports top level objects having superfluous PORs in the output file as below.

```
2 Top level objects found with superfluous extra 'No Group'
PORs in Teamcenter database.
Please review the SFB-Teamcenter-8001528 to evaluate if it is required
to run migrate_propagation_data utility.
1 Top level object found with superfluous duplicate group PORs in
Teamcenter database.
```

Please review the SFB-Teamcenter-8001528 to evaluate if it is required to run migrate_propagation_data utility.

Note:

The number of top level objects listed above will be replaced depending on the number such top level objects existing in the database. For example, if you have 100 top level objects with superfluous PORs that number would then display 100.

Configuration Issue: Non-COTS Multi-value Propagation Attributes.

Migration of PORs with non-COTS multi-value propagation attributes is not supported. Utility reports following information in the output file if such non-COTS multi-value attributes exist.

The following non-COTS multi-value propagation attributes are enabled for propagation:
Migration of PORs will not be supported due to these properties, please contact Siemens Support if you plan to migrate the 'No Group' PORs.
Property: <a3customvlaprop>

ProjectObjectRelation (POR) statistics report file:

This utility also generates a *.report file. This report file contains information about ProjectObjectRelation (POR) statistics. It reports the total number of PORs that exist in the database and the number of PORs for each security group.

Reports the PORs statistics in the output file as below. The POR count varies based on the POR data.

```
ProjectObjectRelation (POR) statistics
Total number of PORs : 204
```

```
POR counts by group
```

```
-----
Security Group I      : 67
Security Group III    : 133
No Group              : 4
```

taxonomy

Generates a character-mode summary of the POM class hierarchy. The taxonomy summary is provided in one of two formats: brief summary or full summary. The brief summary provides a single-line description of each POM class; the full summary every attribute of each POM class.

SYNTAX

taxonomy [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group*] [-b] [-f=*file-name*]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the full path and filename of the password file.

For information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-b

Specifies a brief summary. Each line of the summary provides the following information about a POM class: class depth in the schema, object class name, maximum size in bytes, minimum size in bytes, and application name.

-f

Specifies the name of the output file.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

Localization

find_all_key_value_pairs

Finds all the key/value pairs related to a given installation of Teamcenter. The results are generated in separate text files named *language_locale.txt*, where *language_locale* is the Java standardized name for each locale supported by the system. The result files contain the information using the format *key;value*, where *value* can span over more than one line.

SYNTAX

```
find_all_key_value_pairs [-u=user-id {-p=password | -pf=password-file} -g=group]
-resources_location=path -output_directory=directory [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-resources_location

Specifies the full path to the directory where the Teamcenter localization is located. This value is provided by the **TC_MSG_ROOT** environment variable.

-output_directory

Specifies the directory where the output files are generated.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

ics_localize_class_attributes

Marks class attributes as localizable or nonlocalizable. When marking attributes as nonlocalizable, Teamcenter removes the translations of referenced ICO object properties.

SYNTAX

```
ics_localize_class_attributes [-u=user-id {-p=password | -pf=password-file} -g=group]
-file=file_name -localizable=true|false [-continueOnError] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies a file name in text format. The input file contains the class ID and the attribute ID of the attributes to be marked as localized or nonlocalized. The format of this text file must be as follows:

Class ID|AttributeID_1,AttributeID_2, AttributeID_3...

-localizable

If set to **true**, specifies that the class attributes listed in the input file are marked as localizable. If set to **false**, it marks the specified attributes as nonlocalized and removes the translations of referenced ICO object properties.

-continueOnError

Specifies that the utility continues processing after encountering an error.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
ics_localize_class_attributes -u=Tc-admin-user -p=password -g=group
  -file=my_class_attributes.txt -localizable=true -continueOnError
```

This example marks all the attributes found in **my_class_attributes.txt** as localizable.

my_class_attributes.txt must contain the class and attribute IDs of the attributes to be localized. For example, to mark the **Description** (ID -1200) and **Comments** (ID -1210) attributes contained within the **Components** class (ID FIXT02) as localizable, the contents of the **my_class_attributes.txt** file are:

```
FIXT02 | -1200, -1210
```

I10n_import_export

Exports property values of objects from the database to an XML file for translation purposes. It is also used to import the translated property values from the XML file to the database.

SYNTAX

```
I10n_import_export [-u=user-id {-p=password | -pf=password-file} -g=group]
[-mode= export | import]
-classificationObjectId=class-ID -transferMode=transfer-mode
-type=type-name -properties=list-of-property-names -file=file-name
-noTransExist -locale=locale-name -master -status=localization-status
[-startDate="DD-MMM-YYYY HH:MM"-endDate="DD-MMM-YYYY HH:MM"]
[-savedQueryName=saved-query-name -entryCount=entry-count -entryNames=
entry-names -entryValues=entry-values]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies whether the user is performing an import or export operation. It takes one of the two values, **-import** or **-export**.

-transfermode

Specifies the name of the transfer mode used to export the objects. This transfer mode specifies the traversal rules, filter rules, and property sets to be used for export. It determines what is exported from the system. If not specified, a default transfer mode is used (**TIEUnconfiguredExportDefault**).

-type

Specifies the type for which the instances have to be exported from the database: **bldb0** (view), **icm0** (classification type), **smlb0** (class), **stxt** (key LOV type), **unct_dict** (dictionary attribute).

-properties

Specifies the list of properties (separated by a comma) for which the values must be exported from the database.

-file

Specifies the file name. It acts as an output XML file while exporting objects from the database and acts as an input XML file while importing objects to the database.

-noTransExist

Specifies that no translations exist. This option should only be used with the **-mode=export** option. This option should be used along with the **-locale** option to export objects for a given type that do not have translations for the specified locale.

-locale

Specifies the locale. When this option is used with the **-mode=export** and **-noTransExist** options, it exports all objects for a given type that have no translations for the specified locale. When this option is used with the **-mode=export** and **-master** options, it exports all objects for a given type that have the specified locale as the master locale.

-master

Exports master values to XML file. This option should only be used with the **-mode=export** option.

-status

Specifies the status of the localization. This option is used with the **-mode=export** option to specify the localization status of the property values that must be exported to the XML file for translations. This option is used with the **-mode=import** option to specify the localization status that needs to be set on the property values that are being imported to the database. This option takes one of the following localization statuses: **A** (approved), **P** (pending), **R** (in review), **I** (invalid), or **M** (master).

-startDate

Specifies the start date in the date range using format *DD-MMM-YYYY HH:MM*, for example, **28-OCT-2010 12:01**.

-endDate

Specifies the end date in the date range using format *DD-MMM-YYYY HH:MM*, for example, **28-OCT-2010 12:01**.

-savedQueryName

Specifies the saved query name that already exists in the database. This option can be used with the **-mode=export** option to export the specified list of localizable properties on objects returned by the saved query.

-entryCount

Specifies the entry count of an input to the saved query.

-entryNames

Specifies the list of entry names separated by commas.

-entryValues

Specifies the list of entry values separated by commas.

-classificationObjectId

Specifies the ID of the classification object that needs to be exported.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To search for **Item** objects that have translations so that translations can be done for a new language (for example, Japanese):

```
l10n_import_export -mode=export -type=Item
-l10n=ja_JP -noTransExist -file=abc
```

- To find all **Item** objects that were created in the month of January 2010:

```
l10n_import_export -mode=export -type=Item -file=abc
-startDate=01-JAN-2010 12:01
-endDate=31-JAN-2010 12:01
```

- To import the objects in the Japanese localized file to the database and set the status to pending (P):

```
l10n_import_export -mode=import -file=xyz_ja_JP.xml -status=P
```

- To localize the **object_name** property and description for new items, first create a saved query in the rich client to search for the new items (for example, **findNewItem**) and then export the items using the saved query as an option:

```
l10n_import_export -mode=export -savedQueryName=findNewItem
-type=Item
-properties=object_name,description -file=abc
```

- To export **Item** objects where the master locale of the **object_name** property is Japanese:

```
l10n_import_export -mode=export -type=Item -properties=object_name
-locale=jp_JP -master -file=abc
```

Only one XML file (for the Japanese locale) is created as **output - abc_jp_JP.xml**. All item instances that have the master locale as Japanese for the **object_name** property are included in this file.

- To export **Item** objects where the localization status of the **object_name** property is pending (P):

```
l10n_import_export -mode=export -status=P -type=Item
-properties=object_name -file=abc
```

Only one XML file (for the Japanese locale) is created as output (**abc_jp_JP.xml**). All the item instances that have the master locale as Japanese for the **object_name** property are included in this file.

I10n_purge_translations

Purges the property translations on instances of a given business object. This utility is typically used to clean up the instances of a business object whose property was once marked as localizable with the **Localizable** property constant in the Business Modeler IDE.

For more information, see *Localization process in the Business Modeler IDE* in *Localization using the Business Modeler IDE*.

For example, if you add the **Localization** button to a property by setting the **Localizable** property constant to **true**, and then later decide to remove the button by setting the constant to **false**, you must run the **I10n_purge_translations** utility to remove translations that were entered using the **Localization** button on that property. The utility must be executed after the Business Modeler IDE template is deployed.

The **I10n_purge_translations** utility is necessary only if the **Localizable** constant on a property is moved or deleted from one level, but still exists at another level of the hierarchy. For example, if the **Localizable** property constant is set to **false** on a property on a business object, and there are no sub-business objects that need to be set to **true**, then the utility does not need to be executed. The Business Modeler IDE deploy automatically drops all instances of the **Localization** button translations on the changed property.

SYNTAX

```
I10n_purge_translations [-u=user-id {-p=password | -pf=password-file} -g=group]
-p=properties=business-object:property
-file=file-containing-list-of-properties-to-be-purged
-purge_lot_size=number
-log=file-name
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-properties

Specifies properties to be purged. The format of the property is *business-object:property*, for example, **Item:object_name**. Multiple business object/property combinations can be supplied as a comma-separated string.

-file

Specifies the file to read the *business-object:property* to be purged. Each line must contain a business object/property combination.

-purge_lot_size

Specifies the batch size of instances to be loaded for purging. The default value is 10000.

-log

Specifies the file name to report any failures. The default is the *TC_TMP_DIR/l10n_purge_translations_date.log*. If the **TC_TMP_DIR** environment variable is not set, the log file is created in the */tmp* or *%TEMP%* directory.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

Organization

consumer_license_violation_report

This report consolidates the customer license violation data from individual user log files into a single summary file for the Teamcenter administrator.

User log files (*user-id_license_violation.log*) are created for each user during a Teamcenter session. The summary file, **consumer_license_violation.csv**, contains consolidated information collected from individual users' license violation logs.

» This PC » Local Disk (C:) » Temp » consumer_violations				
Name	Date modified	Type	Size	
 consumer_license_violation.csv	5/22/2020 8:26 PM	Microsoft Excel Comma Separated Values File	1 KB	
 consumer_license_violation_202005211638.csv	5/21/2020 4:34 PM	Microsoft Excel Comma Separated Values File	1 KB	
 consumer_license_violation_202005211758.csv	5/21/2020 4:38 PM	Microsoft Excel Comma Separated Values File	1 KB	
 consumer01_license_violation.log	5/23/2020 3:03 AM	Text Document	1 KB	
 consumer02_license_violation.log	5/22/2020 8:25 PM	Text Document	1 KB	

Consumer license violations are recorded during a session against the user where the user has logged on with the **teamcenter_consumer** access level and performs an operation of a given type that is only allowed for the **teamcenter_author** level. The violations are recorded in a file as the count of violations against a given pair of type (business object) and the operation performed on the given type. The persistent storage, which is a file, is created in the **consumer_violations** directory.

Define the **TC_LOG_DIR** environment variable to indicate where the user violation log file (*user-id_license_violation.log*) is located. If you do not define the **TC_LOG_DIR** environment variable, the violation logs are saved in the **\$TEMP/consumer_violations** directory. If the **consumer_license_violation.csv** file is present in the **\$TEMP/consumer_violations** directory, then the existing copy of the report file is backed up with the name format:
consumer_license_violation_YYYYDDMMHHMM.csv

SYNTAX

```
consumer_license_violation_report [-u=user-id {-p=password|-pf=password-file} -g=group]
-logdir=license-violation-log-directory [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with **dba** privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-logdir

Specifies the name of the license violation log directory, which contains the license violation logs. When this utility is run, these logs are consolidated into the license violation report.

-h

Displays the help for this utility.

EXAMPLES

The following example shows individual user log files for **consumerA** and **consumerB** and the summary file generated by the **consumer_license_violation_report** utility:

- A user with user ID **consumerA** logs on, creates an item (**testItem**), modifies it, and then deletes **testItem**. When **consumerA** logs out of the session, a summary file, **consumerA_license_violation.log**, is created in the **TC_LOG_DIR/consumer_violations** directory. This summary file contains the following data:

```
Item,Create,1
Item,Checkout,1
Item,Checkin,1
Item,Save,3
ItemRevision,Create,1
```

```
ItemRevision,Save,1
Item,Delete,1
ItemRevision,Delete,1
```

- A user with user ID **consumerB** logs on and creates two parts (**PartA** and **PartB**) and an item (**testItem**).

Then, **consumerB** modifies **PartA**, deletes **PartA**, and logs out of the session.

A summary file, **consumerB_license_violation.log**, is created in the **TC_LOG_DIR/consumer_violations** directory. This file contains the following data:

```
Part,Delete,1
Part,Checkin,1
Part Revision,Create,2
Part,Create,2
Part Revision,Save,2
Item,Create,1
Part,Save,4
ItemRevision,Create,1
Item,Save,1
Part Revision,Delete,1
Part,Checkout,1
ItemRevision,Save,1
```

- Run the **consumer_license_violation_report** utility:

```
consumer_license_violation_report -u=Tc_admin -pf=password-file -g=group
-logdir=$TC_LOG_DIR/consumer_violations
```

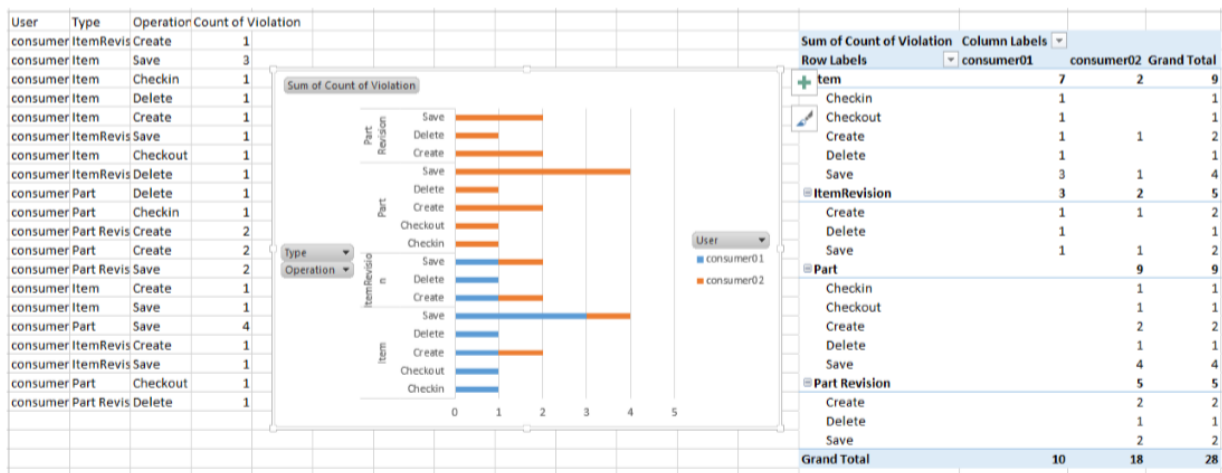
This creates the **consumer_license_violation.csv** file in the location specified by the **-logdir** argument, which in this case is the **\$TC_LOG_DIR/consumer_violations** directory. The file contains the following updated logs:

```
User,Type,Operation,Count of Violation
consumerA,Item,Create,1
consumerA,Item,Checkout,1
consumerA,Item,Checkin,1
consumerA,Item,Save,3
consumerA,ItemRevision,Create,1
consumerA,ItemRevision,Save,1
consumerA,Item,Delete,1
consumerB,Part,Delete,1
consumerB,Part,Checkin,1
```

consumerB,Part Revision,Create,2
 consumerB,Part,Create,2
 consumerB,Part Revision,Save,2
 consumerB,Item,Create,1
 consumerB,Part,Save,4
 consumerB,ItemRevision,Create,1
 consumerB,Item,Save,1
 consumerB,Part Revision,Delete,1
 consumerB,Part,Checkout,1
 consumerB,ItemRevision,Save,1

Note:

You can open and manipulate a .csv file in Microsoft Excel, for example:



> This PC > Local Disk (C:) > Temp > consumer_violations

Name	Date modified	Type	Size
consumer_license_violation	5/21/2020 5:58 PM	Microsoft Excel C...	1 KB
consumer_license_violation_202005211638	5/21/2020 4:34 PM	Microsoft Excel C...	1 KB
consumer_license_violation_202005211758	5/21/2020 4:38 PM	Microsoft Excel C...	1 KB
consumer01_license_violation	5/21/2020 4:29 PM	Text Document	1 KB
consumer02_license_violation	5/21/2020 5:58 PM	Text Document	1 KB

license_server_maintain

Creates, modifies, or deletes a Teamcenter license server.

SYNTAX

```
license_server_maintain [-u=user-id {-p=password|-pf=password-file}
-g=dba] arguments [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

arguments

Specifies one or more of the following:

- **-server_name=license_server_name**

- **-host_name**=*host_name*
- **-port_number**=*port_number*
- **-old_server_name**=*old_server_name*
- **-failover_server_count**=*number_of_failover_servers*
- **-failover_server_list**=*list_of_failover_servers*
- **-function**=*function*

-h

Displays help for this utility.

license_usage

Displays the number of seat-level licenses that are being used for each license server. This information is needed to determine whether unassigned licenses are available, or more licenses need to be purchased in order to create or modify users via the Organization application or the **make_user** utility.

SYNTAX

license_usage [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group*]
arguments [-h]

arguments can include one or more of the following:

- -licenseserver=*license-server*
- -licenselevel=*license-level*
- -all

ARGUMENTS

-u

Specifies the user ID.

This is the user ID of a user with administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the -u and -p arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If this argument is not used, the system assumes the *user-id* value to be the password.

This argument is mutually exclusive with the -pf argument.

-pf

Specifies the **password file**. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-id* value to be the password.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

arguments

Specifies one or more arguments.

-licenseserver=

Specifies the license server, for example **"Default Local License Server"**.

-licenselevel=

Specifies the license level.

Valid values are **author**, **occasionalauthor**, **consumer**, or **admin**.

-all

Specifies usage for all seat-level licenses.

-h

Displays help for this utility.

EXAMPLES

Following are examples of the use of the **license_usage** utility:

- The following example displays the number of licenses used for all seats (author, consumer, occasional author, and admin) on the **Default Local License Server** license server:

```
license_usage -u=Tc_admin -p=password -g=group
               -licenseserver="Default Local License Server" -all
```

A sample report is:

Author:	Purchased Count = 500 Active Users = 436 Inactive Users With Current Month Usage = 5 Available Licenses = 59
Consumer:	Purchased Count = 300 Active Users = 214 Inactive Users With Current Month Usage = 18 Available Licenses = 68
Occasional Author:	Purchased Count = 50

Active Users = 43
 Inactive Users With Current Month Usage = 0
 Available Licenses = 7

Admin:

Purchased Count = 2
 Active Users = 1
 Inactive Users With Current Month Usage = 0
 Available Licenses = 1

- The following example displays author usage information on the **testServer** license server:

```
license_usage -u=Tc_admin -p=password -g=group
               -licenseserver=testServer -licenselevel=author
```

A sample report is:

Author:

Purchased Count = 500
 Active Users = 436
 Inactive Users With Current Month Usage = 5
 Available Licenses = 59

make_user

Creates new users, groups, persons, roles, and volumes outside of a Teamcenter session. This utility also allows you to modify properties of existing user, group, and role objects. The **make_user** utility supports batch mode processing using an input file. Also, you can use this utility as a **replacement for the Idapsync utility** to synchronize your Teamcenter Organization with your Active Directory.

The **make_user** utility creates each of the objects specified by the command line arguments. If the minimum arguments are specified, the utility creates a person and an associated user. If the **-group** argument is supplied, a group is created and the user becomes a member of the group. If the **-role** argument is supplied, a role is created and assigned to that user.

Note:

If a user is created without specifying a role, the user assumes the default role of the group to which they belong. Because a single user can have multiple roles, it is possible for a user to be a member of the same group multiple times, once for each role. Therefore, although the **make_user** utility does not require that a role be specified, it is recommended that one be specified, particularly if there is more than one role associated with the group.

In addition, if a user is created without specifying a password, Teamcenter assigns the user ID as the password.

More than one user can be created at a time. All of the users created become members of the specified group. If both the **-volume** and **-group** arguments are supplied, the group will have a default volume or be granted access to the volume. If any of the specified objects already exist, this utility does not attempt to create them.

If the **-volume** argument is specified, all of the groups that are created are granted access to the volume. If the specified group already exists and does not have a default volume, a volume is designated as the group's default volume.

If the **-os** argument is specified, all groups in the group file are created, and all users in the password file are created and made members of every group to which they belong in the operating system. If the **-group** argument is included with the **-os** argument, only that group is created, and all people who are members of that group are created and added to that group.

If the **-role** argument is specified, all users that are created are assigned that role.

You can modify an existing group, role, person, or user using command line options. Use the **-update** argument to modify the properties of user, person, group, or role objects. Use the **-rename** argument to rename an existing user, person, group, or role. For a user, the **-rename** argument specifies the user ID; for a person, group and role objects, the **-rename** argument specifies the group name and role name, respectively.

Use the **-description** argument to set or modify the description of the group and/or role. When using the **-description** argument with the **-role** and **-group** arguments, the same description is set for both the group and role.

Use the **-volume** argument to create volumes. Before creating volumes, you must have an FMS server cache (FSC) installed and running, and you must set the **FMS_BootStrap_Urls** preference with the FSC host and port information.

Use the **-licenselevel** argument to set the license level when creating a user.

Use the **-datasource** argument to fix an object incorrectly configured as internally or externally managed via LDAP synchronization. Use this argument *only* to correct incorrect synchronization settings. Set this argument to **0** to reset the object as internally managed. Set this argument to **1** to reset the object as externally managed. Set this argument to **2** to reset the object as remotely managed.

Use the **-citizenships** argument to set the citizenships of the user. You can define multiple nationalities, for example, Great Britain and the United States.

Note:

Before using this utility to activate or deactivate group members, you must set the **TC_suppress_inactive_group_members** preference to **0** (zero), which allows system administrators and group administrators to view both active and inactive group members.

This preference determines whether all group members or only active group members are displayed in Teamcenter interfaces. You must set this preference and then relaunch Teamcenter to activate the display suppression for the thin client. In the rich client, you can switch between suppress or display inactive group members using the button located on the **Group** panel of Organization.

SYNTAX

make_user [-u=user-id {-p=password | -pf=password-file} -g=group
arguments [-h]

arguments can include one or more of the following:

- **-allUserDeclaredGeography=geography**
- **-allUserGeography=geography**
- **-citizenships=user-citizenship-list**
- **-cup.custom-property-name=custom-property-value**
- **-defaultgroup=default-group**
- **-defaultlocalvolume=default-local-volume**
- **-defaultrole=default-role**

- **-defaultvolume**=*default-volume*
- **-description**=*description*
- **-discipline**=*discipline-name*
- **-disciplinecurrency**=*discipline-currency*
- **-disciplinedescription**=*discipline-description*
- **-disciplinerate**=*discipline-rate*
- **-file**=*file-name*
- FMS master configuration arguments of which only one can be used:
 - **-fscid***fsc-id*
 - **-filestoregroupid***filestore-group-id*
 - **-loadbalancerid***load-balancer-id*
- **-ga**=0|1|false|true
- **-geography**=*geography*
- **-gm_status**=0|1|false|true
- **-gov_clearance**=*government_clearance*
- **-group**=*group-name*
- **-group_nationality**=*group-nationality*
- **-ignoreIfNotExist**
- **-ip_clearance**=*ip_clearance*
- **-licensebundle**=*license-bundle*
- **-licenselevel**= **author** | **consumer** | **occasionalauthor** ¹
- **-licenseserver**=*license-server*

¹ The types of licenses available depends on your license agreement. For descriptions of the available license levels, see your license agreement documentation.

- **-nationality**=*nationality*
- **-node**=*node-name*
- **-operational_status**=0 | 1 | 2
- **-organization_address**=*organization-address*
- **-organization_alternative_name**=*organization-alternative-name*
- **-organization_id**=*organization-id*
- **-organization_legal_name**=*organization-legal-name*
- **-organization_name**=*organization-name*
- **-organization_type**=*organization-type*
- **-organization_url**=*organization-url*
- **-os**
- **-OSuser**=*name*
- **[-PA1**=*person-PA1-attribute(address-OOTB)*
[-PA2=*person-PA2-attribute(city-OOTB)*
[-PA3=*person-PA3-attribute(state-OOTB)*
[-PA4=*person-PA4-attribute(zip-code-OOTB)*
[-PA5=*person-PA5-attribute(country-OOTB)*
[-PA6=*person-PA6-attribute(organization-OOTB)*
[-PA7=*person-PA7-attribute(GID-OOTB)*
[-PA8=*person-PA8-attribute(mail-code-OOTB)*
[-PA9=*person-PA9-attribute(email-address-OOTB)*
[-PA10=*person-PA10-attribute(phone-number-OOTB)*
- **-parent**=*parent*
- **-password**=*password*
- **-path**=*path-name*
- **-person**=*name*
- **-privilege**=0|1
- **-remove_groupmember**=*group-member*

- **-rename**=*user-id|group-name|role-name|person-name*
- **-role**=*role-name*
- **-security**=*security*
- **-sponsorable**=0|1
- **-status**=0|1
- **-ttc_date**=*ttc-date*
- **-update**
- **-user**=*user-id*
- **-userDeclaredGeography**=*geography*
- **-user_defaultrole**=0|1|false|true
- **-user_denyloginatsites**=*list_of_sites_where_user_is_denied_logon*
- **-user_homesite**=*site-name*
- **-v**
- **-vislicenselevel**=base | standard | professional | mockup
- **-volume**=*name*

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If this argument is not used, the system assumes the *user-id* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the **password file**. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-id* value to be the password.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

arguments

Specifies one or more arguments.

-allUserDeclaredGeography

Sets the user declared geographical location of all users.

Note:

This works for the command line only. This is not supported using an input file.

This argument is processed before other arguments.

-allUserGeography

Sets the geographical location of all users.

Note:

This works for the command line only. This is not supported using an input file.

This argument is processed before other arguments.

-citizenships

Specifies the citizenship of the user. Citizenships are a two-letter country code from ISO 3166. A user can have multiple citizenships. Multiple citizenships are specified with a comma delimiter, such as:

-citizenships=US,GB

-cup.

Sets a user's custom profile attributes.

Note:

The date type format for custom profile attributes is: *yyyy-MM-ddThh:mm:ss zz:zz*; that is, **-cup.custom_property_name=yyyy-MM-ddThh:mm:ss zz:zz**.

- Year: *yyyy*
- Month: *MM*: 1-12 (January = 1)
- Day: *dd*: 1-31
- Hour: *hh*: 0-23
- Minutes: *mm*: 0-59
- Seconds: *ss*: 0-59
- Zulu time, also referred to as Greenwich Mean Time (GMT): *zz*: 0-59

In the following date type format example, the -08:00 is minus 8 hours from GMT (Pacific Standard Time).

-cup.custom_property_name=2014-11-12T14:32:05-08:00

-defaultgroup

Specifies a user's default group. See restriction #6.

-defaultlocalvolume

Specifies the default local volume on the user object. See restriction #8.

-defaultrole

Specifies default role for a group only. See restriction #6.

-defaultvolume

Specifies default volume for group and/or user. If you specify this argument, you must also specify the **-user**, **-password**, and **-group** arguments. See restriction #6.

-description

Specifies the description of a group and/or role.

-discipline

Specifies the discipline, which is a set of users who have a common behavior, for example, developers that have expertise in Linux.

-disciplinecurrency

Specifies the currency for the discipline, for example **USD**, which denotes United States dollars.

-disciplinedescription

Specifies the description of the discipline.

-disciplinerate

Specifies the rate for the discipline.

-file

Specifies that the input file is read to create users or to modify existing users, groups and roles after other arguments are processed. Each record in the file contains the following information:

```
person|user|password|group|role|option_name1|option_value1
      |option_name2|option_value2|...|update
```

option_name is any command line argument and **option_value** is a valid value for **option_name**.

Each field is delimited by the (|) character. The password and role fields can be null (||). The role defaults to the last value specified in either the file or on the command line using the **-role** argument.

Note:

If a password is not specified for the new user, Teamcenter assigns the user ID as the password.

When modifying an existing user, group, or role, specify the properties to be modified by *option_name|option_value* pairs followed by the **update** argument.

FMS master configuration arguments:

Note:

The following three arguments are required if you want the FMS master configuration updated when a volume is created. Do not include these arguments if you intend to update the FMS configuration manually. If supplied, only one of the **fscid**, **filestoregroupid**, or **loadbalancerid** arguments is permitted.

-fscid

Specifies the FSC ID in the FMS configuration to which the volume element is added. The maximum length of this argument is 32 characters.

-filestoregroupid

Specifies the Filestore Group ID in the FMS configuration to which the volume element is added. The maximum length of this argument is 32 characters.

-loadbalancerid

Specifies the load balancer ID in the FMS configuration to which the volume element is added. The maximum length of this argument is 32 characters.

-ga

Specifies the group member admin privilege. If the user is an administrator, set this value to either **1** or **true**. If the user is not an administrator, set this value to either **0** or **false**.

-geography

Specifies the geographical location of the user.

-gm_status

Specifies the group member status. If the group member is inactive, set this value to either **1** or **true**. If the group member is active, set this value to either **0** or **false**.

-gov_clearance

Specifies the level of clearance the user has to classified data.

-group

Specifies the group to which the new user is added. See restriction #1.

-group_nationality

Sets the user group nationality.

-ignoreIfNotExist

Use on upgrade to ignore objects that don't exist.

-ip_clearance

Specifies the intellectual property (IP) clearance level of the user. This is the level of access the user has to sensitive (classified) information.

-licensebundle

Specifies a license bundle for the user.

-licenselevel

Specifies the license level of the user. The default value is **author**.

The types of licenses available depends on your license agreement. For descriptions of the available license levels, see your license agreement documentation.

The Teamcenter administrator can change the licensing level of a user during the month. But, once the user logs on, that user's license is considered reserved for the remainder of the calendar month, even if the user's license level is changed after that date. A single user could thereby unintentionally reserve multiple licenses. For example, a single user could reserve three

licenses during a calendar month (**Author**, **Consumer**, **Occasional Author** license levels) if they have logged on with each license level.

-licenseserver

Assigns the user to the specified license server.

-nationality

Specifies the nationality of the user.

-node

Specifies the network node where the new volume is located. See restriction #3.

-operational_status

Sets a group's operational status. To assign the operational status, set this value to **0** (Active), **1** (Temporarily inactive), or **2** (Permanently inactive).

-organization_address

Sets a group's organization address.

-organization_altername_name

Sets a group's organization alternate name.

-organization_id

Sets a group's organization ID.

-organization_legal_name

Sets a group's organization legal name.

-organization_name

Sets a group's organization name.

-organization_type

Sets a group's organization type.

-organization_url

Sets a group's organization URL.

-os

Specifies that user names and groups specified in the operating system **/etc/passwd** and **/etc/group** files are used to create new users. See restriction #2.

-OSuser

Specifies the operating system user name for the new user. If this argument is not supplied, the operating system user name defaults to the value specified in the **-user** argument.

-PA1

Specifies the person attribute: address.

-PA2

Specifies the person attribute: city.

-PA3

Specifies the person attribute: state.

-PA4

Specifies the person attribute: zip code.

-PA5

Specifies the person attribute: country.

-PA6

Specifies the person attribute: organization.

-PA7

Specifies the person attribute: GID (group ID).

-PA8

Specifies the person attribute: mail code.

-PA9

Specifies the person attribute: e-mail address.

-PA10

Specifies the person attribute: phone number.

-parent

Specifies a group's parent group. See restriction #6.

-password

Creates a password for the new user.

Note:

If a password is not specified for the new user, Teamcenter assigns the user ID as the password.

-path

Specifies the full path to the location of the new volume. See restriction #3.

-person

Specifies the person associated with the new user.

-privilege

Specifies a group's privilege. See restriction #5.

-remove_groupmember

Specifies the removal of the existing user from the specified group and role.

Note:

You cannot use any other action arguments with the **-remove_groupmember** argument. For example, you cannot use the **-update** argument with the **-remove_groupmember** argument. You must use the **-update** argument separately.

-rename

Specifies a new name for an existing user, group, role, or person.

-role

Specifies the role to which the new user is assigned.

-security

Specifies a group's security.

-sponsorable

Specifies a user is sponsorable.

If set to **1** (true), the user is sponsorable and can access Teamcenter with a sponsoring user. The default is **0** (zero), which indicates the user is not sponsorable.

-status

Specifies a user's status. See restriction #7.

-ttc_date

Specifies the Technology Transfer Certification (TTC) date, which is the date when the user's qualification for viewing data marked as government classified lapses.

The date type format is: *dd-Mon-yyyy*. For example, 03-Jan-2018.

-update

Use when modifying any of the existing user, group, or role objects. See restriction #4.

-user

Creates a new Teamcenter user.

-userDeclaredGeography

Specifies the current geographical location of the user.

-user_defaultrole

Specifies default role for all users. To assign the default role flag, set this value to either **1** or **true**. To remove the default role flag, set this value to either **0** or **false**.

-user_denyloginatsites

Specifies the sites for which the user is denied logon. When specifying more than one site, use a comma separator. For example: site1,site2,site3.

-user_homesite

Specifies the home site for the specified user.

-v

Runs the utility in verbose mode to display the maximum amount of information. Typically, non-verbose utility sessions only display error messages.

-volume

Specifies the new volume to be created. See restrictions #3 and #9.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. If the command line argument parameter contains spaces, it must be enclosed in quotes, for example, "**product validation**". This restriction does not apply to arguments from a file using the **-file** argument because quotes are automatically added using this argument.
2. The **-os** argument is only valid on Linux platforms.
3. Volume creation requires **-volume**, **-node** and **-path** arguments, except when creating a new cloud volume. Also, the **-volume**, **-node**, and **-path** arguments must not be separated by other arguments.
4. Only one object (user, group, or role) can be updated at a time.

If the **-user**, **-group**, or **-role** arguments are specified when using the **-update** argument, the user object is assumed as the target object of the update.

If the **-group** or **-role** arguments are specified without the **-user** argument, the group object is updated.

If the **-role** argument is specified without the **-user** or **-group** arguments, the role is updated. The object to be updated must already exist.

5. The privilege setting for a group can be either **0** or **1**. **1** implies that the group has DBA privileges. **0** implies a non-DBA group. The default value is **0**.

6. The objects specified for the **-defaultgroup**, **-defaultrole**, **-defaultvolume**, and **-parent** arguments must be existing objects.
7. The valid values for the **-status** argument are **0** (active) and **1** (inactive). If these values are not specified, the default status is active.
8. If both **defaultlocalvolume** and **defaultvolume** arguments are specified, their values must be different. The default local volume must be configured as a valid FMS volume.
9. To create volumes, an FSC must be installed and running, and the **FMS_Bootstrap_Urls** preference must be set with the FSC host and port information.
10. If a user with the same group/role combination is a member of any project, the user is not removed and the system displays an error message.
11. If a user is a member of only one group and role and there is an attempt to remove group membership, the user is not removed and the system displays an error message.

RETURN VALUES

Return value upon success 0

Return value upon failure 1

TIPS FOR SYNCHRONIZING YOUR ORGANIZATION OBJECTS IN TEAMCENTER WITH LDAP DATA

Because the **ldapsync** utility is obsolete, Siemens Digital Industries Software recommends you use the **make_user** utility in batch mode to create and update your organization objects in Teamcenter. You are responsible for creating and populating a file to use in batch mode, which involves probing your existing LDAP and formatting it using the correct syntax for the **make_user** utility to read.

1. Determine which fields are populated in LDAP and extract the information.
2. Format the data for a batch file, as described in the **-file** arguments description.

For example, to update group administrator status for the existing three users (**tom**, **dan**, and **bob**), whose user IDs are (**tm**, **dm**, and **bp**), create a file named **makeuserinput_ldap_to_tc.lst** containing the following data:

```
|tm||Engineering|Designer|ga|true|update
|dm||Engineering|Designer|ga|true|update
|bp||Engineering|Designer|ga|true|update
```

Each field is delimited by the (|) character.

3. Enter the following command on a single line.

```
make_user -u=Tc-admin-user -p=password -g=group  
-file=makeuserinput_ldap_to_tc.lst
```

EXAMPLES

Create User...	Command Line
Specifying new group/role/person	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=tm -group=london.dev -role=planner -person=tom</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre>tom tm london.dev Planner</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying existing group/role/person for new user	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm -group=london.dev -person=dan -role=planner</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre>dan dm london.dev Planner</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying only existing group	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm1 -group=london.dev -person=dm1</pre>
The default role is considered while creating the new user.	To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre>dm1 dm1 london.dev </pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying default volume	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=bp -group=london.dev -person=bob -defaultvolume=volume</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data:

Create User...	Command Line
	bob bp1 london.dev defaultvolume volume 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying sponsorable	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=bp -sponsorable=1 -update</pre>
Specifying all organization properties	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm2 -group=london.dev -role=planner -person=dm2 -ga=true -defaultvolume=volume -licenseserver=28000_pncitcflex2 -licensebundle=COMMSITE -licenselevel=consumer -user_homesite="IMC--1758036214" -user_denyloginatsites="IMC--1759825534" -vislicenselevel=standard</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre>dm2 dm2 cambridge.dev designer defaultvolume volume defaultgroup cambridge.dev user_defaultrole true licenseserver 28000_pncitcflex2 licensebundle COMMSITE licenselevel author ga true user_homesite IMC--1758036214 user_denyloginatsites IMC--1759825534 vislicenselevel standard</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying all ADA properties (user)	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm3 -group=london.dev -role=planner -person=dm3 -ip_clearance=super-secret -gov_clearance=secret -geography=IN -nationality=IN -citizenships=IN -userDeclaredGeography=IN -ttc_date=30-JUN-2023</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre>dm3 dm3 london.dev planner ip_clearance secret gov_clearance secret geography IN nationality IN citizenships IN userDeclaredGeography IN ttc_date 30-JUN-2023</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>

Create User...	Command Line
Specifying all ADA properties (group)	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -group=pune.dev -organization_name=splm2 -organization_legal_name=splm_legal2 -organization_alternate_name=splm_all2 -organization_address="baner, balewadi, pune-24" -organization_url="http://www.siemens.com" -organization_id=002 -group_nationality=US -organization_type=1</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> pune.dev organization_name splm organization_legal_name splm_legal2 organization_alternate_name splm_alt2 organization_address "baner, balewadi, pune-24" organization_url http:// www.siemens.com organization_id 002 group_nationality US organization_type 1</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying all person properties	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm4 -group=london.dev -role=planner -person=dm4 -PA1="10824 Hope Street" -PA2=Cypress -PA3=California -PA4=90630-5214 -PA5=USA -PA6="Siemens PLM Software" -PA7=ZAPARED5 -PA8=dm4 -PA9=dm4@splm.siemens.com -PA10=+1(714)9526223</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre>dm4 dm4 london.dev planner PA1 "10824 Hope Street" PA2 Cypress PA3 California PA4 90630-5214 PA5 USA PA6 "Siemens PLM Software" PA7 ZAPARED5 PA8 dm4 PA9 dm4@splm.siemens.com PA10 +1(714)9526223</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying new group/role/ person with space in name	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=dm5 -group="Electrical Engineering -role="Electrical Engineer" -person="Donald McIntire"</pre> To execute this command using the -file argument:

Create User...	Command Line
	- Create an input file, user.lst, containing the following data: <pre>Donald McIntire dm6 Electrical Engineering Electrical Engineer</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Update User...	Command Line
Adding to a new group	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=tm -group=cambridge.dev -role=planner -person=tom</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre>tom tm cambridge.dev planner update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Changing role	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -user=tm -group=cambridge.dev -role=designer -person=tom</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre>tom tm cambridge.dev designer update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Changing default group	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=tm -defaultgroup=cambridge.dev</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> tm defaultgroup cambridge.dev update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>

Update User...	Command Line
Setting default role	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=tm -group=cambridge.dev -role=designer -user_defaultrole=true</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> tm cambridge.dev designer user_defaultrole true update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Setting group administrator	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=tm -group=cambridge.dev -role=designer -ga=true</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> tm cambridge.dev designer ga true update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying all organization properties	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=tm2 -group=cambridge.dev -role=planner -defaultgroup=cambridge.dev -user_defaultrole=true -ga=true -defaultvolume=volume -licenseserver=28000_pncitcflex2 -licensebundle=COMMSITE -licenselevel=consumer -user_homesite="IMC--1758036214" -user_denyloginatsites="IMC--1759825534" -vislicenselevel=standard</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> tm cambridge.dev designer defaultvolume volume defaultgroup cambridge.dev user_defaultrole true licenseserver 28000_pncitcflex2 licensebundle COMMSITE licenselevel author ga true user_homesite IMC--1758036214 user_denyloginatsites IMC--1759825534 vislicenselevel standard update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>

Update User...	Command Line
Specifying all ADA properties (user)	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=dm3 -ip_clearance=top-secret -gov_clearance=top- secret -geography=GB -nationality=GB -citizenships=GB -userDeclaredGeography=GB -ttc_date=30-JUN-2025</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> dm3 ip_clearance top-secret gov_clearance top- secret geography GB nationality GB citizenships GB userDeclaredGeography GB ttc_date 30-JUN-2023 update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying all ADA properties (group)	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -group=london.dev -organization_name=splm2 -organization_legal_name=splm_legal2 -organization_alternate_name=splm_all2 -organization_address="baner, balewadi, pune-24" -organization_url="http://www.siemens.com" -organization_id=002 -group_nationality=US -organization_type=1</pre> To execute this command using the -file argument: - Create an input file, user.lst, containing the following data: <pre> london.dev organization_name splm2 organization_legal_name splm_legal2 organization_alternate_name splm_alt2 organization_address "baner, balewadi, pune-24" organization_url http:// www.siemens.com organization_id 002 group_nationality US organization_type 1 update</pre> - Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Specifying all person properties	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=dm3 -group=london.dev -role=planner -person=dm3 -PA1="10824 Hope Street" -PA2=Cypress -PA3=California -PA4=90630-5214 -PA5=USA -PA6="Siemens PLM Software" -PA7=ZAPARED5 -PA8=dm4 -PA9=dm4@splm.siemens.com -PA10=+1(714)9526223</pre> To execute this command using the -file argument:

Update User...	Command Line
	1. Create an input file, user.lst, containing the following data: <pre>dm3 PA1 "10824 Hope Street" PA2 Cypress PA3 California PA4 90630-5214 PA5 USA PA6 "Siemens PLM Software" PA7 ZAPARED5 PA8 dm5 PA9 dm5@splm.siemens.com PA10 +1(714)9526223 update</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>
Setting user status to inactive	<pre>make_user -u=tcadmin -p=tcadmin -g=dba -update -user=dm3 -status=1</pre> To execute this command using the -file argument: 1. Create an input file, user.lst, containing the following data: <pre> dm3 status 1 update</pre> 2. Enter the command using the -file argument: <pre>make_user -u=tcadmin -p=tcadmin -g=dba -file=user.lst</pre>

Teamcenter reporting

install_default_report_designs

Creates the default report designs contained in the **default_report_design.xml** file as part of the Teamcenter installation process. Other report designs can be imported into the database by adding them to the **default_report_design.xml** file and rerunning the utility.

SYNTAX

```
install_default_report_designs [-u=user-id {-p=password | -pf=password-file} -g=group] -file=xml-file-list -design_name=report-design-name -create_new [-update_all | -update_query | -update_pff | -update_formatter [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the name of the XML file containing report design definitions.

-design_name

Specifies the name of the report design.

-create_new

Creates new objects.

This cannot be used with update modes.

-update_all

Updates all existing objects (query, pff, and formatter).

-update_query

Updates existing query objects.

-update_pff

Updates existing pff objects.

-update_formatter

Updates existing formatter objects.

-v

Displays detailed status information.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

RETURN VALUES

Return value upon success	0
Return value upon failure	>1, -1

EXAMPLES

To create default report enter the following command on a single line:

```
install_default_report_designs -u=user_name -p=user_name -g=dba
-file=$TC_DATA\report_writer\default_report_designs.xml
```

XML FILE FORMAT

The format required for the XML file is as follows.

```
<ReportDesign>
  <DesignName>Admin - Objects By Status</DesignName>
  <DesignDesc>This report returns objects of a specified type
released to a specified status.</DesignDesc>
  <Query>
    <QueryName>Admin - Objects By Status</QueryName>
    <QueryDesc>This query was created to support the Admin -
Objects By Status Report.</QueryDesc>
    <QueryClass>WorkspaceObject</QueryClass>
    <QueryClause>SELECT qid FROM WorkspaceObject WHERE
      "object_type" = "${Type = ItemRevision}" AND
      "release_status_list.name" = "${Release Status
= }"
    </QueryClause>
    <DomainFlag>QRY_DOMAIN_LOCAL</DomainFlag>
  </Query>
  <Pff>
    <PffName>Admin - Objects By Status</PffName>
    <PffDesc>This PFF was created to support the Admin -
Objects By Status Report.</PffDesc>
    <PffClass>WorkspaceObject</PffClass>
    <PffClause>
      WorkspaceObject.object_name;Object Name,
      WorkspaceObject.object_type;Object Type,
      WorkspaceObject.release_status_list.name;Release Status,
      WorkspaceObject.date_released;Date Released,
      WorkspaceObject.owning_user.user_id;User Name,
      WorkspaceObject.owning_group.name;Group Name
    </PffClause>
  </Pff>
  <Formatter>
    <Filename>default_xml_template.xsl</Filename>
    <Datasettype>XMLReportFormatter</Datasettype>
  </Formatter>
  <Formatter>
    <Filename>default_excel_template.xlt</Filename>
```

```
        <Datasettype>ExcelReportFormatter</Datasettype>  
    </Formatter>  
</ReportDesign>
```

XML file format

rep_batch_report

Used in batch or shell script files to generate reports in batch mode when the user selects the **Batch mode** option in the **Generate ME Report** dialog box.

The following administrative tasks must be performed to enable batch reporting:

1. Create a report request flat file containing default values and details of item ID, revision ID, operating system report location, Teamcenter report location, report format, revision rule, variant rule, transfer mode, and status. The format of the file is as follows:

```

_____Start of format_____
start_global_definitions
User ID          user_name
Password         user_name
TC_ROOT          w:\iman_wnti
TC_DATA          w:\src\iman\data
Default Report Location Teamcenter # possible values are OS or Teamcenter
Default OS Location   C:\temp\Reports
Default Revision Rule Latest Working
Default Saved Variant Rule DemoVariantRule
Default Transfer Mode web_reports
Default Formatter     : Product_Structure.xml
Delimiter          ~
end_global_definitions
start_data_definition
#ItemID~Revision~Report Location~OS Location~Revision Rule~Saved Variant Rule~
Transfer Mode~Formatter~Status
000234~A~OS~~~MyVariantRule~~Standard Product~
end_data_definition

start_data_definition
#ItemID~Revision~Report Location~OS Location~Revision Rule~Saved Variant Rule~
Transfer Mode~Formatter~Root Product Tag~ (line continued)
Root Product Rev rule~Root Product Var rule~Root process Tag ~
Hierarchy UID tags~level~Root PlantTag~Root Plt Rev Rule~Root Plt Var Rule~Status
GM00071~001~OS~~Latest Working~~web_reports~Product_Structure.xml~~~~~
ABC000007~A~OS~~Latest Working~~web_reports~station_weld.xml~SvNJ1puVl9P7VD~
(line continued)
Latest Working~~htNJmIl8l9P7VD~x5FJmIl8l9P7VD,BKKJmIl8l9P7VD,RaDJmIl8l9P7VD
~3~zwDJ2_1$l9P7VD~Latest Working~~
end_data_definition
_____End of format_____

```

Note:

A sample batch file and shell script file are located in the **TC_ROOTweb/htdocs/web_reports/data** directory. In addition, you can manually append data to an existing **batch_request** file and run the utility.

2. Schedule a task in the operating system.
3. Select the program and specify the execution date and time.

The utility performs the following activities:

1. Reads the location of the report request flat file from the value of the **Batch_Report_Request_File** preference.
2. Reads the global definition values.
3. Parses each line in the flat file, checks the status field of the line, and processes the line if the status is not **success**.
4. An XML file is generated for each line in the file, using the revision rule, variant rule, and closure rule associated with the transfer mode. If the rules are not specified in the line, default rules are used.
5. The report format (style sheet) is applied on the XML file and report HTML files are generated. The datasets are exported during the transformation.
6. If the report location is specified as Teamcenter, the dataset is created and the files generated are attached to the item revision, including the exported dataset files.
7. If the report location is specified as OS, the reports are saved at the OS location specified in the file.
8. Create or update the log file for the process.
9. Update the status field once the line is processed.

SYNTAX

rep_batch_report [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group*] -h

ARGUMENTS

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

import_export_reports

Allows report definitions, their dependent data (for example, saved query definitions and property set definitions), and their associated style sheets to be exported from one Teamcenter server and imported to another Teamcenter server.

SYNTAX

```
import_export_reports {-import | -export | -execute}  
[-u=user-id -p=password | -pf=password-file -g=group]  
-stageDir=directory -reportId=report-identifier -overwrite  
[-reportFile=xml-output-file] [-f=xml-file] [-h]
```

ARGUMENTS

-import

Specifies the report definitions are to be imported to the Teamcenter server.

-export

Specifies the report definitions are to be exported from the Teamcenter server.

-execute

Generates a report in the command line. This argument requires the **-f** argument.

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-stageDir

Specifies the fully qualified name of the directory that contains all of the report definitions and its associated data in predefined format.

Note:

If you use **-reportId** to import a single report, you must place the report definition file and resource file in a folder with the same name as the report ID and place it in the directory and folder specified by the **-stageDir** argument.

-reportId

Specifies the ID of the report definition.

-overwrite

Overwrites the existing report definitions during import.

If you want to modify the closure rule, change the transfer mode name and the closure rule name in the respective XML file.

Note:

Use this argument with caution. If you have done any customizations to the commercial-off-the-shelf data (COTS) templates, that data is overwritten by this argument.

-reportFile

Specifies the name of the XML containing the list of report templates.

Note:

If you provide both **-reportId** and **-reportFile**, **-reportId** is used to import a single report and **-reportFile** is ignored.

-f

Specifies the name of an XML file containing report parameters. This argument is used with the **-execute** arguments.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following command exports report definitions and associated data (style sheets) to the file system pointed by the **-stageDir** argument:

```
import_export_reports -export -u=<username> -p=<password> -g=<group>
    -stageDir=<data directory> -reportId=<reportname>
```

- The following command imports report definitions from a Teamcenter server:

```
import_export_reports -import -u=<username> -p=<password> -g=<group>
    -stageDir=<data directory> -reportId=<reports>
```

- The following is an example of how to create an input file format and run the **-execute** command with the **-f** filename option to generate a report:

You can import or export report definitions by using a sample XML file. For more information about sample XML files, see *TC_DATA\crflcrf_item_report_offline_execution_sample.xml* for an item report and *TC_DATA\crflcrf_summary_report_offline_execution_sample.xml* for a summary report.

1. Create a **sumsap.xml** input file format as follows:

Summary report example:

```
<?xml version="1.0" encoding="UTF-8"?>
<CrfReport>
  <SummaryReport command="execute" id="TC_2007_00_SUM_RPT_0001"
    stylesheet_name="admin_ownership_html.xsl"
    output_report_file="D:\report builder\sumrpt.html">
    <report_parameter name="Name" value="test"/>
    <report_parameter name="Type" value="ItemRevision"/>
    <report_option name="report_locale" value="en_US" />
  </SummaryReport>
</CrfReport>
```

Item report example:

```
<?xml version="1.0" encoding="UTF-8"?>
<CrfReport>
  <ItemReport command="execute" id="TC_2007_00_ITM_RPT_0003"
    stylesheet_name="ps_bom_report_excel.xsl"
    output_report_file="D:\report builder\itmrpt.html" >
    <context_item name="000004" rev="A" />
    <report_option name="report_locale" value="en_US" />
    <report_option name="rev_rule" value="Revision_Rule_Name" />
    <report_option name="variant_rule" value="Variant_Rule_Name" />
  </ItemReport>
</CrfReport>
```

Revision rule name and variant rule name are optional. If the revision rule name is not specified, the default is **Latest Working**.

2. To generate the report, execute the input file using the following command:

```
Import_export_reports -u=user_name -p=password -g=dba -execute -f="d:\report
builder\sumsap.xml"
```

The system generates the **sumrpt.html** report in the **d:\report builder** directory.

Note:

If you do not specify the **output_report_file** option, the report is generated in the path specified in the **TC_TMP_DIR** environment variable.

If you do not set the **TC_TMP_DIR** variable, then the system creates the report in the **C:\temp** directory for Windows or **/tmp** directory for Linux systems.

Teamcenter interface

mgradmin

Utility for managing the Teamcenter server manager database.

- Update the obfuscated password the Server Manager uses to connect to the database.
- Display the contents of the database respective to the given Server Manager Pool.
- Clean database table entries for the given Server Manager Pool.

SYNTAX

mgradmin [-h] [-password] [-list [PoolID]] [-clean [PoolID]]

ARGUMENTS

-h

Displays help for this utility.

-password

Update the obfuscated password the Server Manager uses to connect to the database.

Note: This does not change the database password, only the configured value that the Server Manager uses to connect to the database.

-list

Display the contents of the database respective to the given Server Manager Pool. If the PoolID is not provided, the full database contents (for this cluster) are listed.

-clean

Clean database table entries for the given Server Manager Pool. If the PoolID is not provided, all database tables (for this cluster) are removed from the database.

Column configuration utilities

delete_uiconfig

Deletes column configuration objects.

SYNTAX

```
delete_uiconfig [-u=user-id] { [-p=password | -pf=password-file] } [-g=group] (args)
```

ARGUMENTS

This is a pre-upgrade utility to delete column configuration objects and their related **BusinessObject** instances. Execute the utility by using one argument at a time. If multiple arguments are used, then only one argument is considered and the rest are ignored. The order of precedence is as follows.

1. **-prefix**
2. **-client_scope**
3. **-column_config_id**
4. **-column_config_name**
5. **-col_def_obj_type**

Only administrative users are allowed to run the utility and is not intended to be used as a general purpose utility.

-u

Teamcenter user ID.

-p

Teamcenter password.

-g

Teamcenter group.

-prefix

Deletes all client scopes and column configurations whose ID starts with the specified prefix. **ColumnDef** objects referencing the column configuration are also deleted.

-client_scope

Deletes the client scope, its associated column configurations, and the **ColumnDef** objects referenced by the client scope.

-column_config_id

Deletes the column configuration object specified by this ID. Multiple values can be provided using comma separators.

-column_config_name

Deletes the column configuration object specified by this name. Multiple values can be provided using comma separators.

-col_def_obj_type

Deletes the **ColumnDef** objects referencing the specified type. Multiple values can be provided using comma separators.

-log

Full path to write execution results. If the log file option is not specified, then the default log file will be created in the **TEMP** directory.

-h

Displays the help for this utility.

export_uiconfig

Exports column configuration XML files.

SYNTAX

export_uiconfig [-u=*user-id*] { [-p=*password* | -pf=*password-file*] } [-g=*group*] (args)

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If **-u** is used without a value, the operator system user name is automatically applied.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the path and file name for the output file.

-for_group

Specifies the group or groups to which user interface configuration objects are exported.

To specify more than one group, use commas to separate the group names.

-for_role

Specifies role or roles to which user interface configuration objects are exported.

To specify more than one role, use commas to separate the role names.

-for_workspace

Specifies workspace or workspaces to which user interface configuration objects are exported.

To specify more than one workspace, use commas to separate the workspace names.

-client_scope_URI

Specifies, for export only, the column configurations and command contributions corresponding to the indicated client scope only.

The **Awb0OccurrenceManagement** scope must be added if you use this argument.

Note:

Client scope refers to a sublocation in the client, not group or role.

-client

Specifies, for export only, the column configurations and command contributions corresponding to this client only.

-h

Displays help for this utility.

EXAMPLES

To export all the site-wide column configurations:

```
export_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml
```

To export all the column configurations specific to the **Engineering** group:

```
export_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml  
-for_group=Engineering
```

To export the inbox table column configuration specific to the **Engineering** group.

```
export_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml  
-for_group=Engineering -client_scope_URI=fnd0Inbox
```

import_uiconfig

Imports column configuration XML files.

SYNTAX

import_uiconfig [-u=*user-id*] { [-p=*password* | -pf=*password-file*] } [-g=*group*] (args)

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If **-u** is used without a value, the operator system user name is automatically applied.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the path and file name for the input file.

-for_group

Specifies the group or groups to which user interface configuration objects are imported.

To specify more than one group, use commas to separate the group names.

-for_role

Specifies role or roles to which user interface configuration objects are imported.

To specify more than one role, use commas to separate the role names.

-for_workspace

Specifies workspace or workspaces to which user interface configuration objects are imported.

To specify more than one workspace, use commas to separate the workspace names.

-action=<value>

override (*default*): Specifying this option will override the existing column configuration with the new one.

skip: Specifying this option will cause the existing column configuration to be retained and will not be updated. However, if there is no existing configuration, then one will be created.

merge: Specifying this option will merge the new column configuration with the existing one. This may cause column order to change.

-h

Displays help for this utility.

Note:

If neither **-for_group** or **-for_role** arguments are included, user interface configuration objects are imported with **Site** scope.

EXAMPLES

To import the configuration specified in the XML file for multiple roles, in this case: **Designer** and **engineer**.

```
import_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml
-for_role=Designer,engineer
```

To import the configuration specified in the XML file for the **Engineering** group.

```
import_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml
-for_group=Engineering
```

To import the configuration specified in the XML file for multiple groups, in this case **Engineering**, **system** and **"Validation Administration"**.

```
import_uiconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml
-for_group=Engineering,system,"Validation Administration"
```

Note:

A parameter containing spaces, such as **Validation Administration** in the preceding example, must be enclosed in double quotation marks ("").

Workspace utilities

export_wsconfig

Exports workspace configuration XML files.

SYNTAX

```
export_wsconfig [-u=user-id] { [-p=password | -pf=password-file] } [-g=group] (args)
```

ARGUMENTS

-u

Specifies the user ID.

A user with administration privileges is used as the value name for the user ID. If **-u** is used without a value, the operator system user name is automatically applied.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-for_workspace=<workspace-name>

Specifies workspace or workspaces which are exported.

To specify more than one workspace, use commas to separate the workspace names.

-file

Specifies the path and file name for the output file.

-h

Displays help for this utility.

EXAMPLES

To export all the workspace configurations:

```
export_wsconfig -u=<user_id> -p=<password> -g=dba -file=exported.xml
```

To export the workspace configuration for the **myCustomWorkspace** workspace:

```
export_wsconfig -u=xxx -p=xxx -g=dba -for_workspace=myCustomWorkspace -file=exported.xml
```

import_wsconfig

Imports workspace configuration XML files.

SYNTAX

```
import_wsconfig [-u=user-id] { [-p=password | -pf=password-file] } [-g=group] (args)
```

ARGUMENTS

-u

Specifies the user ID.

A user with administration privileges is used as the value name for the user ID. If **-u** is used without a value, the operating system user name is automatically applied.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action=<value>

Currently, the only action available is **delete**.

The **delete** option will remove existing workspace configurations.

-file

Specifies the path and file name for the input file.

-h

Displays help for this utility.

EXAMPLE

To import the workspace configurations specified in the XML file:

```
import_wsconfig -u=<user_id> -p=<password> -g=dba -file=ws.xml
```




3. Product configuration utilities

Content management

contmgmt_migration_90

If you extend your data model by adding customized topic classes or properties, and you are migrating from a Teamcenter 8.x version, update the custom topic classes or properties using the **contmgmt_migration_90** utility.

For more information, see *What is Content Management?* in *Content Management — Deployment and Administration*.

SYNTAX

```
contmgmt_migration_90 [-u=user-id -p=password | -pf=password-file -g=group]  
[-cp]  
[-ap]  
[-dryrun]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-cp

Specifies the .txt file containing the class names.

-ap

Specifies the .txt file containing the property names.

-dryrun

Generates a report of the changes to be made but does not update the database.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To test to see which custom topic classes or properties are updated:

```
contmgmt_migration_90 -u=Tc-admin-user -p=password -g=group  
-cp=classNames.txt  
-ap=attributes.txt dryrun
```

- To update the custom topic classes and properties:

```
contmgmt_migration_90 -u=Tc-admin-user -p=password -g=group  
-cp=classNames.txt  
-ap=attributes.txt
```

contmgmt_migration_1123

In Teamcenter 11.2.3 and later, the relationship type between content references and graphic content references is changed from a GRM (generic relationship management) relation to a **PSOccurrence** relation. If you are upgrading from any prior release, you must run the **contmgmt_migration_1123** utility to update these relationships.

For more information, see *What is Content Management?* in *Content Management — Deployment and Administration*.

SYNTAX

```
contmgmt_migration_1123 [-u=user-id -p=password -g=group]
[-log=file-name-path]
[-errorlog=file-name-path][-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

-p

Specifies the password.

-g

Specifies the group associated with the user.

-log

Specifies the path to the log file. If a path is not specified, Teamcenter creates a log at a default location.

-errorlog

Specifies the path to the error log. If a path is not specified, the errors appear in the standard migration log

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter.*

RESTRICTIONS

None.

contmgmt_migration_100

In Teamcenter 10 and later, a new relationship exists between the **PSOccurrence** object between two related topics and the reference topic type that is between the two topic types. When upgrading from a prior release, the **contmgmt_migration_100** utility must be run to add this relationship to **PSOccurrence** objects. If more than one reference type exists between the two topic types, the utility selects one and the selection is noted in a log file.

For more information, see *What is Content Management?* in *Content Management on Rich Client — Usage*.

SYNTAX

```
contmgmt_migration_100 [-u=user-id -p=password | -pf=password-file -g=group]
[-clear] [-dryrun] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-clear

Clears any existing **Ctm0RefTopicTypeR** relations before processing.

-dryrun

Generates a report of the changes to be made but does not update the database.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To run a test to see which **PSOccurrences** objects will have relationships added:

```
contmgmt_migration_100 -u=Tc-admin-user -p=password -g=group -dryrun
```

- To update all the **PSOccurrences** objects with the new relationship:

```
contmgmt_migration_100 -u=Tc-admin-user -p=password -g=group
```

Product structure maintenance

bom_expand

DESCRIPTION

Expands the bill of materials (BOM) and generates a report on the BOM structure and expansion statistics.

You can use this utility to find the total size and depth of a BOM structure as well as the number of lines at each level of the structure.

Additionally, you can use this utility to assign in-context IDs to each line in a structure. This is necessary when comparing structures using the accountability check in manufacturing.

This utility provides a **bomset** mode that can help you expand the BOM in equal-sized sets without encountering memory issues.

SYNTAX

```
bom_expand [-u=user-id -p=password | -pf=password-file -g=group]
-item=item | -key=item-key
[-revision_rule=revision-rule] [-rev=revision]
[-saved_variant_rule=saved-variant-rule]
[-use_packing] [-show_unconfigured] [-show_unconfigured_variants]
[-props=comma-separated-list-of-property-names] [-level_props]
[-output=output-filename] [-props= | -props_output=
comma-separated-list-of-property-names]
[-bom_report_file=report-file-name] [-report_mode= [TOP | LEVEL | FULL]]
[-no_verbose] [-mem_debug] [-bom_closure_rule=closure-rule]
[-bomset] [-bomset_criteria=[MEMORY | NUMLINES | PERCENT]]
[-bomset_help]
[-bomset_criteria_data=[Number-of-Lines | PERCENT]]
[-bomset_override_data_complexity_preference=[true | false]]
[-bomset_use_mem_percentage_preference=percent]
[-create_absoccs]
[-create_absocc_ids] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with or without Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the item ID for which the associated BOM view revision (BVR) is traversed. (BVRs are associated with revisions corresponding to the specified item.)

-key

Specifies the key for which the associated BVR is traversed. (BVRs are associated with revisions corresponding to the specified item.)

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-revision_rule

Specifies the revision rule to use when configuring the BOM window.

If this argument is omitted, uses the default revision rule, **Latest Working**.

-rev

Specifies the item revision specified by the **-item** argument. The revision must have an associated BVR.

-saved_variant_rule

Specifies the variant rule to apply to the BOM window when configuring the BOM.

-use_packing

Indicates whether the packed lines in the BOM window should be packed or unpacked.

-show_unconfigured

Indicates whether unconfigured BOM lines are considered for BOM expansion.

-show_unconfigured_variants

Indicates whether to consider unconfigured BOM lines due to variant conditions for BOM expansion.

-props

Specifies the set of properties to fetch for BOM structure lines. If the value specified is **RAC**, fetches all properties required by the rich client. You can also specify a comma-separated list of properties. For example, if you want properties **bl_formula** and **bl_variant_state**, specify this argument as **-props= bl_formula, bl_variant_state**.

-level_props

Indicates whether to fetch all properties together for a BOM level during expansion. If this flag is absent, fetches properties once for the entire BOM after expansion.

-output

Directs the output to a specific file by specifying a file name.

-props_output

Lists BOM line properties separated by a comma that should be written to output. The **-props** and **-props_output** arguments are mutually exclusive.

-bom_report_file

Specifies the report file name, for example, **report.csv** (CSV file type is preferred).

-report_mode

Specifies report generation mode. Valid values are:

- **TOP** prints expansion report for top node of BOM only.
- **LEVEL** prints summary of expansion for all levels in BOM.
- **FULL** prints detailed expansion report for each node in BOM.

The default mode is **LEVEL**.

-no_verbose

Indicates whether to write detailed output to the console.

-mem_debug

Indicates whether to print additional debug information related to memory usage during BOM expansion.

-bom_closure_rule

Specifies BOM closure rule to apply to BOM window during BOM expansion.

-bomset

Indicates whether expansion should be done in equal-sized sets of BOM lines.

-bomset_criteria

Specifies creation criteria of BOM sets for expansion. Valid values are:

- **MEMORY**

The BOM is divided and expanded in sets that occupy the same memory. This memory is calculated based on the available system physical memory and BOM size.

- **NUMLINES**

The BOM is divided and expanded in sets that contain the same number of BOM lines. Use the argument **-bomset_criteria_data** to specify the number of lines for each BOM set.

- **PERCENT**

The BOM is divided and expanded in sets that contain the number of lines which are the same percent as the total number of BOM lines in the BOM. Use the argument **-bomset_criteria_display** to specify this percent.

The default is **MEMORY**.

-bomset_help

Prints examples for using **bomset** options.

-bomset_criteria_data

Specifies data for BOM set creation to use with **-bomset_criteria**. Valid values are **NUMLINES** or **PERCENT**.

This argument should be used when **-bomset_criteria_data=NUMLINES** or **-bomset_criteria_data=PERCENT**.

For example, if you have a 10,000 item BOM and you want it to be divided and expanded in sets each having the size 1,000 items, include the following arguments in your command:

- **-bomset -bomset_criteria=NUMLINES**
- **-bomset_criteria_data=1000**

If you want a 10,000 item BOM to be divided and expanded in sets that are 10 percent of its size, include the following arguments in your command:

- **-bomset -bomset_criteria=PERCENT**
- **-bomset_criteria_data=10**

-bomset_override_data_complexity_preference

Specifies whether BOM lines have complex data, such as incremental changes and absolute occurrences, which can decide the size of the BOM set when the **-bomset_criteria** value is set to **MEMORY**. The valid values are **true** or **false**. The default value is **false**.

-bomset_use_mem_percentage_preference

Specifies the percent of system memory available to use for expansion. Valid values must be set between **0** and **100**.

-create_absoccs

Creates absolute occurrence appearance path node (APN) objects on a BOM structure during expansion.

-create_absocc_ids

Creates or uses existing absolute occurrence IDs on a BOM structure during expansion.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

This utility can generate different output files based on the **[-output=output-filename]** or the **-bom_report_file** options.

RESTRICTIONS

This is a debug utility and should only be used for debugging purposes.

EXAMPLES

- BOM expansion with properties, using revision rules and variant rules.

bom_expand

```
-u=user-id -p=password -pf=password-file -item=item
-rev=revision -revision_rule=revision-rule -saved_variant_rule=
saved-variant-rule -bom_closure_rule=closure-rule
-props=RAC -level_props
```

- BOM expansion with out properties, using revision rule and variant rule.

```
bom_expand -u=user-id -p=password -pf=password-file -item=item -rev=revision
-revision_rule=revision-rule -saved_variant_rule=
saved-variant-rule
```

- BOM expansion with properties, using revision rule and variant rule and closure rule.

```
bom_expand -u=user-id -p=password -pf=password-file -item=item
-rev=revision -revision_rule=revision-rule -saved_variant_rule=
saved-variant-rule -props=RAC -level_props bom_report_file=report-file-name
-report_mode=FULL
```

- BOM expansion with properties, using revision rule and variant rule and providing output in a file (txt).

```
bom_expand -u=user-id -p=password -pf=password-file -item=item
-rev=revision -revision_rule=revision-rule -saved_variant_rule=
saved-variant-rule -props=RAC -level_props -output=output-filename
```

- BOM expansion with properties, using revision rule and variant rule and closure rule with properties, using revision rule and variant rule and providing BOM report file (csv) (BOM data).

```
bom_expand -u=user-id -p=password -pf=password-file -item=item
-rev=revision -revision_rule=revision-rule -saved_variant_rule=
saved-variant-rule -props=RAC -level_props bom_report_file=report-file-name
-report_mode=FULL
```

- BOM expand with BOM set.

```
bom_expand -u=user-id -p=password -item=item_id -bomset
bom_expand -u=user-id -p=password -item=item_id -bomset
-bomset_criteria=PERCENT
-bomset_criteria_data=% of bomlines in a set.
Value between 0-100>
bom_expand -u=user-id -p=password -item=item_id -bomset
-bomset_criteria=NUMLINES
-bomset_criteria_data=num of lines in a set
bom_expand -u=user-id -p=password -item=item_id -bomset
-bomset_override_data_complexity_preference=true/false
-bomset_use_mem_percentage_preference=% of system mem
which can be used for bomset/expansion. Value between 0 -100.
```

Only valid when -bomset_criteria=Empty value of MEMORY

bom_roll_up_report

Creates BOM properties rollup reports. The reports can be created systematically by a task scheduler to generate weekly or daily reports to track property changes of assembly structures.

SYNTAX

```
bom_roll_up_report [-u=user-id -p=password | -pf=password-file -g=group]  
{[-item=item | [-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2...,keyAttrN=keyValN]]]}  
[-rev=revision] [-revrule=revision-rule]  
[-effdate=mm:dd:yyyy:HH:MM:SS | now | today]  
[-varrule=variant-option] [-name=name] [-desc=description]  
-template=name:scope-context [-folder] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the item to be used as the root line for the BOM properties rollup report.

-key

Specifies the key to be used as the root line for the BOM properties rollup report. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev

Specifies the item revision to be used as the root line for the BOM properties rollup report. If this argument is omitted, the report is based on the latest revision of the item.

-revrule

Specifies the revision rule to use when creating the BOM properties rollup report. If this argument is omitted, the default revision rule, **LATEST_WORKING**, is used.

-effdate

Specifies the date to use when configuring the effectivity of the assembly structure. If this argument is omitted, no effectivity date is set.

- *mm* specifies the month (01–12)
- *dd* specifies the day (01–31)
- *yyyy* specifies the year (0001–9999)
- *HH* specifies the hour (00–23)
- *MM* specifies the minute (00–59)
- *SS* specifies the second (00–59)

-varrule

Specifies the variant option set to use when setting the variant options of the assembly structure. If this argument is omitted, default variant options are used or no options are set.

-name

Specifies the name of the BOM properties rollup report. If this argument is omitted, an auto-generated name is created.

-desc

Contains a description of the BOM properties rollup report. If this argument is omitted, an autogenerated description is created. The autogeneration occurs only if a default description is defined in the BOM properties rollup template that was used to create the report.

-template

Specifies the name and scope context of the BOM properties rollup template to use when creating the BOM properties rollup report. Scope context is the user, group, or site scope identifier.

-folder

Indicates that the system is to place the new BOM properties rollup report dataset into the users **Newstuff** folder. If user privileges do not allow for BOM properties rollup report datasets to be attached to item revisions, the report is placed in the user's **Newstuff** folder.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following example creates a BOM properties rollup report with autologin, autogeneration options, and attaches the report to the item revision:

```
bom_roll_up_report -item=12345678 -template="masstemplate:engineering"
```

- The following example creates a BOM properties rollup report with autologin, autogeneration options, and attaches the report to the **Newstuff** folder:

```
bom_roll_up_report -item=12345678 -template="masstemplate:engineering"
-folder
```

- The following example creates a BOM properties rollup report with autologin, but with no autogeneration options:

```
bom_roll_up_report -item=12345678 -rev=B -name="My Report"
-desc="Validating mass values." -template="masstemplate:engineering"
```

- The following example creates a BOM properties rollup report with autologin and configures the assembly:

```
bom_roll_up_report -item=12345678 -rev=B -revrule="Released"  
  -effdate=02:20:2006 -varrule="High performance option set"  
  -name="My Report" -desc="Validating mass values."  
-template="masstemplate:engineering"
```

bomwriter

Emits a bill of materials (BOM), in a variety of file formats, to a nominated file optionally restricted to selected areas of the BOM.

For example, you can use this utility to export product structure information from Teamcenter to a PLM XML file that can be consumed by several applications. If the size of the product structure is very large (several thousand occurrences) you may run the export overnight. In such situations, consider using the **PLMXML_sdk_threshold** preference to minimize memory consumption during the export.

For more information about using this preference to serialize PLM XML objects, therefore reducing memory consumption during large exports, see the *Teamcenter Environment Variables*.

SYNTAX

```
bomwriter [-u=user-id {-p=password | -pf=password-file} -g=group]
[-noprompt] [-h | -help]
{-bookmark=bookmark-file |
-item_list=input-file-name |
-item=item-name | -key=item-key}
[-rev=revision]
[-selected=input-file-name]
[-subselected=input-file-name]
[-output_file=output-file-name]
[-revision_rule=configuration-rule]
[-show_substitutes=true | false]
[-show_unconfigured=true | false]
[-show_unconfigured_occ_eff=true | false]
[-show_variants=true | false]
[-view=view-type-name]
[-descendants=true | false]
[-flatten=true | false]
[-packed_window=true | false]
[-transient_unpack=true | false]
[-smstring=true | false]
[-svrule=saved-variant-rule-name [:rev-id:rule-id]]
-format=format where format is one of the following:
    index
    psup
    plmxml
    ajt
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-noprompt

Specifies that if autologin fails, the system will not prompt for an interactive login. This is a standard autologin option.

-help

Displays help for this utility.

-bookmark

Specifies a simple bookmark file from which to extract the default root item ID, root revision ID, and revision rule (overruled by any revision rule specified on the command line). The parsing of this file is primitive: one complete XML element per line is expected.

-item_list

Specifies a list of item IDs, one per line, with optional output file names on the same line. If no file name clause is provided, the default is **itemid.format** without **+** options. These items are processed successively.

-item

Specifies the item ID for the root of the BOM structure.

-key

Specifies a string used to identify an item instead of specifying an item using the **-item** and **-rev** arguments.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev

Specifies the item revision for the root of the BOM structure.

-view

Specifies the view to be used.

-output_file

Specifies the output file. The **stdout** file is the default.

-format

Specifies the output file format with optional modifiers, for example:

-format=ajt+native+asm_jt_file

-format=index

Format used with the **-selected** or **-subselected** option. No modifiers.

-format=psup

psup format with the **+props=xxx** modifier representing comma-separated BOM line properties. You can also specify a delimiter using the **delimiter** option. The default delimiter is a comma (,).

For example, to specify the semicolon as a delimiter:

-format=psup+delimiter=;

+props=xxx

Comma-separated BOM line properties are exported.

+delimiter=x

Single-character delimiter is used to separate each column in the output.

-format=plmxml

plmxml format with the following modifiers:

+strict

Minor errors are fatal.

+type= xxx

Use nondefault builder.

+tmode= xxx

Use nondefault transfer mode.

+transform=[None | Cumulative | Absolute]

Use specified **transformType** on Occurrence.

+locales=language_codes

Specifies the languages for the export. Separate multiple languages by commas (for example, **en_US, fr_FR**). The language IDs follow the standard locale naming conventions (for example, **en_US**). If no locales are specified for export, the database scalar value (attribute master) is exported to PLM XML scalar fields.

+ua= xxx

User attributes specifier.

The syntax is a combination of a target specifier followed by a key specifier followed by a property specifier, all separated by commas. For example:

```
-format=plmxml+ua=target:<target name>,  
key:<key name>,prop:<prop name>...
```

The target, key, and property values are controlled by the **INTEROP_ExtraPLMXMLInstanceAttributes** preference. If there is more than one key/property value specified for a given target, you can specify each without repeating the **target** keyword. The properties are exported as **UserValue** under the **UserData** element under the **target** element.

In the following example, the **UserData** element appears under the **Occurrence** element as it is the target.

```
<UserValue value="prop_value" title="key"></UserValue>
```

In the following example, the user exports the **bl_rev_owning_user** property of **BOMLine** under the **Occurrence** element with a title of **OwningUser**. The same applies to **bl_rev_owning_group**.

```
-format=plmxml+ua=target:Occurrence,  
key:OwningUser,prop:bl_rev_owning_user,  
key:OwningGroup,prop:bl_rev_owning_group
```

+revid_off

Turn off revision ID in the PRV name attribute.

+varuid_on

Turn on variant UID.

+grdvua_on

Turn on user attributes for GRDVA in instance element.

-format=ajt

ajt format with the following modifiers:

+nt

ajt file attribute in Windows format. For example:

d:\folder\tk0404c2_mod_5q8050016xwq6.jt

+unix

ajt file attribute in UNIX format. For example:

/folder/tk0404c2_mod_5q8050016xwq6.jt

+native

ajt file attribute in machine-native (Linux or Windows) format.

+uidtag

ajt file attribute in **uidtag** format. For example:

BVHRD95\$LV1P\$n\$hD.jt

+identity

Causes a missing transform to become an identity transform rather than a fatal error.

+strict

Minor errors are fatal.

+skip_fake_part

Skips dummy part for subassemblies.

+asm_jt_file

Outputs the associated JT files (if any) information for any intermediate lines in the assembly.

-selected

Specifies the input file to nominate particular lines as selected, defaults to the root line if none is selected. If the specified input file is empty, the output should contain configuration information only (no BOM lines), where supported.

Run **bomwriter** once with the **-f=index** argument, edit the resulting file to remove lines that should not be selected, and run **bomwriter** again with the same parameters, but **-f** to the format you want and **-s** indicating the edited **-f=index** file.

Note:

Review the index output file carefully before using it as the input for this option. If the output contains any special characters, for example: new line, the **-selected** option may not function properly.

-subselected

Specifies the subselected items file. Edit **-f=index** for format.

-descendants

Specifies whether descendants of selected lines should be included in the output, using **true** or **false** values. Defaults to **true**. Selected lines and ancestors of selected lines are always included.

-flatten

Presents the selected lines as a tree in which the root node is the immediate parent of all selected lines. The transforms of the selected lines are combined with the transforms of their ancestor's lines to compensate for their disappearance. Valid values are **true** and **false**. Defaults to **false**.

The **-flatten** argument reverses the default for the **-descendants** argument, because the **-flatten** argument never needs descendent lines.

-packed_window

Use a packed BOM window. Valid values are **true** and **false**. The default value is **false**.

Do not use with the PLM XML format.

-transient_unpack

Use transient unpacking. Valid values are **true** and **false**. The default value is **false**.

Do not use with the PLM XML format.

-smstring

Use **smstring** output, printed to **stdout**. Valid values are **true** and **false**. The default value is **false**.

-revision_rule

Specifies a named revision rule. Defaults to the site default, frequently the **latest working** revision.

-show_substitutes

Specifies that substitutes be shown. Defaults to the site default value.

-show_unconfigured

Specifies that unconfigured lines be shown. Defaults to the site default value.

-show_unconfigured_occ_eff

Specifies that lines unconfigured by occurrence effectivity be shown. Defaults to the site default value.

-show_variants

Specifies that unconfigured variants be shown.

Defaults to the site default value. However, if **-svrule** is specified, the default value is assumed to be **false**, unless you explicitly specify **true** in the command line entry.

-svrule

Specifies a saved variant rule to use when configuring the exported BOM. Optionally specify a particular revision of a variant rule object by providing its revision ID and rule ID separated by colons. Variant rule revision use is supported for only **Cfg0VariantCriteria**, a revisable subclass of the **VariantRule** object.

-setpcamode

Sets Product Configurator variant mode on a BOM window. This mode is set on the window only when the configurator context is associated with the top item or the input variant rule is Product Configurator variant based. If neither of these is true and the user wants to export the sub-assembly's Product Configurator variant-based variant formula property, then use this command line argument. For a migrated structure, both classic and Product Configurator variant data are present. Typically sub-assemblies will not have a configurator context associated. In that case, the user should use this command line argument to export the Product Configurator variant-based variant formula.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Run the **bomwriter** utility with the following arguments to write an **AJT** file for compilation (using **asciitojt**) on the current platform where some, but not all, BOM lines have transform matrices:

```
bomwriter -bookmark=somefile.bkm -format=ajt+native+identity
-output_file=somefile.ajt
```

- Run the **bomwriter** utility with the following arguments to write selected parts of a BOM window in **AJT** format (without descendants). First, create the index file from which to select parts:

```
bomwriter -item=XYZ001 -format=index -output_file=xyz001.index
```

- Edit the **selected.index** file to remove various lines and then run the utility as follows:

```
bomwriter -item=XYZ001 -format=ajt+native -selected=xyz001.index  
-descendants=false -output_file=xyz001.ajt
```

- Output associated JT files information for any intermediate lines for item XYZ001.

```
bomwriter -item=XYZ001 -format=ajt+native+asm_jt_file  
-output_file=xyz001.index -rev=A
```

- The following command calls the **bomwriter** utility with PLM XML format output on **item1** and produces the **Export_WithTranslations.plmxml** file. It also writes the **bl_item_object_desc** BOM line property as **Item_Desc UserData** under the **Occurrence** element. The French and German translations of the localized properties on **item1** are also exported to **Text** elements.

```
bomwriter -u=user -p=password -g=group -item=item1  
-output_file=Export_WithTranslations.plmxml  
-format=plmxml+ua=target:Occurrence, key:"Item_Desc", prop:  
"bl_item_object_desc"+locales=fr_FR, de_DE
```

cfg0_gen_enforced_cond

Generates enforced conditions for configurator rules if no enforced condition already exists. An enforced condition is a system-generated expression that comprises the model, applicability, and payload expressions of a configurator rule. The presence of enforced conditions on configurator rules results in better performance when the Product Configurator validates variant configurations.

Enforced conditions were introduced in Teamcenter 10.1.4. If you authored configurator rules in earlier Teamcenter versions, those configurator rules do not have enforced conditions, and you should consider running this utility once to provide faster validations. Likewise, if you are migrating from an earlier Teamcenter version, run this utility to create enforced conditions for all migrated rules.

Syntax

```
cfg0_gen_enforced_cond [-u=user-id -p=password | -pf=password-file -g=group]  
-output=dirname [-batch_size=value] [-h]
```

Arguments

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-output

Specifies the absolute path to the generated log file. If no path is specified, the report is generated in the current working directory of the application.

-batch_size

Specifies how many rules should be processed in a batch. If no batch size is specified, the default value of 100 rules is used.

-h

Displays help for this utility.

Restrictions

This utility must be run by an account with system administration privileges.

Examples

- To generate enforced conditions with the default batch size of 100 rules:

```
cfg0_gen_enforced_cond -u=<user> -p=<password> -g=dba  
-output=c:\temp\enfcond.log
```

- To generate enforced conditions with a batch size of 1000 rules:

```
cfg0_gen_enforced_cond -u=<user> -p=<password> -g=dba  
-batch_size=1000 -output=c:\temp\enfcond.log
```

cleanup_arr_absoccddata

Allows a user to find and report bad absolute occurrence data that was created during a **Save As** operation. The bad absolute occurrence data is created if the item being copied has assembly arrangement data associated with it.

Note:

Only privileged users may run this utility.

SYNTAX

```
cleanup_arr_absoccddata -u=user-ID {-p=password | -pf=password-file} [-g=group]
[report]
[destroy_item=itemID ]
[destroy_all]
[-findDataWithInvalidQual]
[-removeDataWithInvalidQual]
[-output=output_file ]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-report

Searches the entire database and reports all bad absolute occurrence data.

-destroy_item

Destroys bad data for one item ID that was created by a **Save as** operation. The **-item** parameter is mandatory for this argument; otherwise, the utility fails.

-destroy_all

Destroys all bad absolute occurrence data found in the database. If the **-item** parameter is provided, the utility fails.

-findDataWithInvalidQual

Searches the entire database and output UIDs of absolute occurrence data with invalid or stubbed absolute occurrence data qualifiers at a replica site.

-removeDataWithInvalidQual

Destroys all absolute occurrence data with invalid or stubbed absolute occurrence data qualifiers at a replica site.

-output

Contains information about bad absolute occurrence data during report modes. If not specified, the **cleanup_arr_absoccddata.txt** default file is used. For destroy modes, the report indicates whether destroy is successful. In the output file, the following information will be dumped: saved as item ID, saved as item name, original item ID, original item name, and UID of bad absolute occurrence data.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

cleanup_orphan_arrangements

cleanup_orphan_arrangements is a tool to find and destroy orphan arrangements. Arrangements in Teamcenter are typically managed and published by NX. Arrangements are published to Teamcenter by creating a GRM relationship with the assembly BVR (BOM view revision) and creating referential relationships from higher level assemblies to lower level assembly arrangements. An arrangement is

configured by the arrangement selection at the parent level and then at the next level, all the way to the top of the assembly.

Note:

Only privileged users may run this utility.

Recommended steps for using this utility:

1. Run the utility in report mode only.
2. Check the report to see if the BVR has more than one arrangement.
3. Back up the database.
4. Run the utility in destroy mode for one UID or selected UIDs.
5. Run the utility for the entire database.

Syntax

```
cleanup_orphan_arrangements -u=user-ID {-p=password | -pf=password-file} [-g=group]
[-report]
[-searchForArrs][-searchForQuals][-destroyAll][-destroySelected]
[-output=output_file]

[-inputUIDs=input_uids_file]
[-arr=input_uid]

[-h]
```

Arguments

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p or -pf

Specifies the user's password or a password file. These arguments are mutually exclusive.

For more information about managing password files, see *Manage password files*.

-g

Specifies the group associated with the user.

-report

Searches the entire database to find orphan arrangements and puts the results in the *–output* file.

-searchForArrs

Search for orphan arrangements.

-searchForQuals

Search for qualifiers that point to arrangements from other BVRs.

-destroyAll

Destroy all orphan arrangements.

-destroySelected

Destroy selected arrangements provided using one UID (*-arr*) or a file (*-inputUIDs*) containing UIDs.

-output

Contains information about bad arrangements or arrangement qualifiers in report mode. If not specified, the utility uses the default **cleanup_orphan_arrangements.txt** file. In destroy modes, the report indicates whether the destroy action is successful.

The following information is dumped in the output file:

- Arrangement name.
- UID of arrangement.
- Last saved date of arrangement.
- Arrangement's BVR name.

-inputUIDs

The file containing all UIDs of the arrangements to destroy. Each line of the text file contains one arrangement UID.

-arr

The arrangement UID to destroy.

-h

Displays help for this utility.

Environment

As specified in *Setting up a Teamcenter command-line environment*.

Files

As specified in *Log files produced by Teamcenter*.

Restrictions

None.

Examples

- Discover phase

```
cleanup_orphan_arrangements -report -output=output.txt --searchForQuals
```

Output:

```
=== Begin to look for Qualifiers that point to arrangement from wrong
BVR ===
```

```
Performing query on DB to find Qualifiers that point to arrangements
from wrong BVR
```

```
End of Performing Query
```

```
Found 24 bad qualifiers
```

```
UID of arrangement
```

```
SDStrF8KLQIfvB , Qualifier is not destroyed
```

```
COWtrRRSLQIfvB , Qualifier is not destroyed
```

```
CiTtLrkCLQIfvB , Qualifier is not destroyed
```

```
QqRt7eFgLQIfvB , Qualifier is not destroyed
```

```
QqftrF8KLQIfvB , Qualifier is not destroyed
```

```
QuStrF8KLQIfvB , Qualifier is not destroyed
```

```
S9ctLrkCLQIfvB , Qualifier is not destroyed
```

```
SlbtrF8KLQIfvB , Qualifier is not destroyed
```

```
S6atrF8KLQIfvB , Qualifier is not destroyed
```

```
wCbtrF8KLQIfvB , Qualifier is not destroyed
g2Wt7eFgLQIfvB , Qualifier is not destroyed
iYftrRRSLQIfvB , Qualifier is not destroyed
iXdtrRRSLQIfvB , Qualifier is not destroyed
ieVtrRRSLQIfvB , Qualifier is not destroyed
ihetrF8KLQIfvB , Qualifier is not destroyed
weat7eFgLQIfvB , Qualifier is not destroyed
AHet7eFgLQIfvB , Qualifier is not destroyed
yiYtrF8KLQIfvB , Qualifier is not destroyed
ysdt57wLLQIfvB , Qualifier is not destroyed
y0atL1SSLQIfvB , Qualifier is not destroyed
y3dtrRRSLQIfvB , Qualifier is not destroyed
y_QtrRRSLQIfvB , Qualifier is not destroyed
zIatrRRSLQIfvB , Qualifier is not destroyed
zZatrRRSLQIfvB , Qualifier is not destroyed
```

- **Fix Phase**

```
cleanup_orphan_arrangements -output=ouput.txt -destroySelected -inputUIDs=bad_qual.txt -
searchForQuals
```

The input file **bad_qual.txt** is used to collect a list of arrangements to destroy:

```
=== Begin to look for Qualifiers that point to arrangement from wrong
BVR ===
```

```
Performing query on DB to find Qualifiers that point to arrangements
from wrong BVR
```

End of Performing Query

Found 24 bad qualifiers

UID of arrangement

SDStrF8KLQIfvB , Qualifier is destroyed successfully

COWtrRRSLQIfvB , Qualifier is destroyed successfully

CiTtLrkclQIfvB , Qualifier is destroyed successfully

QqRt7eFgLQIfvB , Qualifier is destroyed successfully

QqftrF8KLQIfvB , Qualifier is destroyed successfully

QuStrF8KLQIfvB , Qualifier is destroyed successfully

S9ctLrkclQIfvB , Qualifier is destroyed successfully

SlbtrF8KLQIfvB , Qualifier is destroyed successfully

S6atrF8KLQIfvB , Qualifier is destroyed successfully

wCbtrF8KLQIfvB , Qualifier is destroyed successfully

g2Wt7eFgLQIfvB , Qualifier is destroyed successfully

iYftrRRSLQIfvB , Qualifier is destroyed successfully

iXdtrRRSLQIfvB , Qualifier is destroyed successfully

ieVtrRRSLQIfvB , Qualifier is destroyed successfully

ihetrF8KLQIfvB , Qualifier is destroyed successfully

weat7eFgLQIfvB , Qualifier is destroyed successfully

AHet7eFgLQIfvB , Qualifier is destroyed successfully

yiYtrF8KLQIfvB , Qualifier is destroyed successfully

ysdt57wlLQIfvB , Qualifier is destroyed successfully

y0atL1SSLQIfvB , Qualifier is destroyed successfully

```

y3dtrRRSLQIfvB , Qualifier is destroyed successfully
y_QtrRRSLQIfvB , Qualifier is destroyed successfully
zIatrRRSLQIfvB , Qualifier is destroyed successfully
zZatrRRSLQIfvB , Qualifier is destroyed successfully

```

```

=== End of search and cleanup for Orphan Arrangements ===

```

cleanup_psdata

Cleans up obsolete and duplicate product structure data. This utility has three options:

- **Clean up duplicate variants**

The **duplicate_variants** option identifies and optionally removes duplicate variants from the database. Duplicate variants occur when two variant objects have the same name and are owned by the same item. Duplicate variants can cause the importing or exporting of variants to fail. To avoid this and any other potential problems that may arise due to duplicate variants, Siemens Digital Industries Software recommends cleaning up duplicate variants.

- **Clean up absoccViewQualifier objects**

The **absoccViewQualifier** option identifies and optionally removes unreferenced **absoccViewQualifier** objects from the database.

- **Repair markup changes**

The **repairMarkupChanges** option updates the markup changes in the database. The way Mass Updates stores data has changed for some cases. This option aligns the existing data with the new way.

Note:

You must run the utility with this option after upgrading Teamcenter from a previous release.

Note:

You must have administrative privileges to run this utility.

Syntax

```
cleanup_psdata -u=user-ID {-p=password} [-g=group]
[-duplicate_variants]
[-absoccViewQualifier]
[-repairMarkupChanges][-report_file_path=path]
[-auto_cleanup]
[-output=path]
[-report]
[-destroy_all]
[-h]
```

Arguments

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user's password.

-g

Specifies the group associated with the user.

-duplicate_variants

Specifies the option to clean up duplicate variants.

-absOccViewQualifier

Specifies the option to clean up **absoccViewQualifier** objects.

-report_file_path

Specifies the output file path when using the **duplicate_variants** option. Set this to a valid local drive path for the report files.

-auto_cleanup

Specifies to automatically delete duplicate variants when found.

-output

Specifies the output file path when using the **absoccViewQualifier** option. Set this to a valid local drive path for the report files.

-report

Specifies to report unreferenced **absOccViewQualifier** objects found in the database. The report file is saved in the location specified by the **output** argument.

-destroy_all

Specifies to remove all unreferenced **absOccViewQualifier** objects found in the database.

-h

Displays help for this utility.

-repairMarkupChanges

Updates the markup changes in the database.

Environment

This utility should be run in a shell where the Teamcenter environment is set up.

Files

A report file is generated in the specified location.

Restrictions

None.

Examples

- **Duplicate variants report**

Report only

To generate a duplicate variants report:

```
cleanup_psdata [-u=user-name] [-p=password] [-g=group-name]
-duplicate_variants -report_file_path=path
```

This command generates a file named **duplicate_variants_report_timestamp.csv** in the location specified in the **report_file_path** argument. The report lists all duplicate variant objects with UIDs of variant options, variant revisions, numbers of expressions referring to duplicate options, and other information.

Report and clean up

To generate a duplicate variants report and automatically clean up duplicate variant objects:

```
cleanup_psdata [-u=user-name] [-p=password] [-g=group-name]
-duplicate_variants -report_file_path=path auto_cleanup
```

This command generates a file named **duplicate_variants_report_timestamp.csv** in the location specified in the **report_file_path** argument. The report lists all duplicate variant objects with UIDs of variant options, variant revisions, numbers of expressions referring to duplicate options, and other information. The report also lists variants that were successfully cleaned up.

- **absoccViewQualifier objects report**

Report only

To generate a report of all unreferenced or invalid **absoccViewQualifier** objects:

```
cleanup_psdata [-u=user-name] [-p=password] [-g=group-name]
-absoccViewQualifier -output=path -report
```

This command generates a report file in the location specified in the **output** argument.

Report and clean up

To generate a report and automatically remove all unreferenced or invalid **absoccViewQualifier** objects from the database:

```
cleanup_psdata [-u=user-name] [-p=password] [-g=group-name]
-absoccViewQualifier -output=path destroy_all
```

This command generates a report file in the location specified in the **report_file_path** argument. The report lists all **absoccViewQualifier** found and indicates which objects were successfully cleaned up.

- **Repair markup changes**

To update the markup changes in the database:

```
cleanup_psdata -[u=user-name] -p=[password] -g=[group-name]
-repairMarkupChanges
```

This command searches for all markup changes in the database that contain **PSOccurrence** and replaces it with **PSOccurrenceThread**.

create_or_update_bbox_and_tso

This utility performs the following tasks:

- Creates, updates or deletes bounding box data from NX (**UGMASTER**) and JT (**DirectModel**) datasets.
- Creates or updates TruShape data (**.tso** files) for JT files.

In a multisite environment, this utility creates the **.tso** files only for datasets that belong to the owning site. It does not create the **.tso** files for replica datasets.

- Generates reports of:
 - NX or JT datasets not having a bounding box object.
 - NX datasets not having a **UGPartBBox** form.
 - JT datasets not having a **.tso** file. Using this report, the same utility can generate Dispatcher requests so the translation (conversion to the bounding box and TSO) occurs on the dedicated Dispatcher machine.

SYNTAX

```
create_or_update_bbox_and_tso [-u=user-id -p=password | -pf=password-file
-g=group] -mode=usermode -translation_mode=operatingmode -generate_ets_request
-delete_all_bboxes -dataset=dataset_uids -dataset_list=filename
[-objects=object_list-object_list=filename]
[-scope=ALL | PRODUCT] [-product=product_item_id
-output_dir=dirname [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies one of the following modes:

- **query**

Generates a report of NX or JT datasets that require updates to the bounding boxes, TSO files (JT files only), or missing **UGPartBBox** forms (NX datasets only).

- **process**

Creates or updates bounding boxes, TSO files (for JT datasets only) or both for a set of NX or JT datasets.

- **query+process**

Generates a report of NX or JT datasets that require updates to the bounding boxes or TSO files (JT files only). It then creates or updates bounding boxes or TSO files (for JT datasets only) for the identified datasets.

- **delete**

Deletes the specified datasets.

-translation_mode

Specifies one of the following translation modes:

- **JTTOBBOX**

Creates or updates the bounding boxes in JT datasets.

- **JTTOTSO**

Creates or updates the TSO files in JT datasets.

- **JTTOBBOX+JTTOTSO**

Creates or updates the bounding boxes and TSO files in JT datasets.

- **NXBBOXTOBBOX**

Creates or updates the bounding boxes in NX datasets.

- **processAll**

Use with the **NXBBOXTOBBOX** mode to force creation of bounding boxes for all NX datasets.

- **NXBBOXFORM**

Creates a report listing all NX datasets for which the associated **UGPART-BBOX** form is not available.

-generate_ets_request

Specify this argument when you specify **query** mode and are working only with JT datasets. Creates a Dispatcher request in the database for each JT dataset that needs updates to bounding boxes or TSO files. Before you use this argument, ensure the Dispatcher translation service is configured and running, as described in *Using Dispatcher*.

-delete_all_bboxes

Deletes all the bounding boxes, multibounding boxes and relations between them from the database. (It does not delete TruShape data and you should re-create the TSO files if necessary.)

-dataset

Specifies one or more dataset UIDs as a string separated with commas, in the format:

ds1, ds2, ..., dsn

This argument is valid only if you specify **process** mode.

-dataset_list

Specifies the absolute path to an input file that contains a list of dataset UIDs to process. Each dataset must appear on a new line of this file. This argument is valid only if you specify **process** mode.

-objects

Specifies a comma-separated list of UIDs of primary objects whose bounding boxes you want to delete, in the format *ob1,ob2,...,obn*. This argument should be used only in delete mode.

-object_list

Specifies the fully qualified path name of a file containing a list of UUIDs of primary objects whose bounding boxes you want to delete. Each UUID should appear on a new line. This argument should be used only in delete mode.

-scope

Limits the scope of queries to the accountability table. Specify either **ALL** (consider all products in the accountability table) or **PRODUCT** (consider items listed under the product specified by the **-product** argument).

-product

Specifies the ID of a product item that is listed in the accountability table.

-output-dir

Specifies the absolute path to the directory where the report file is generated. If no path is specified, the report is generated in **./output_dir**. This argument is valid only if you specify **query** mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Create bounding box and TruShape data for all the JT datasets in the database. You can do this in one of two ways:
- Create a report file of the datasets that do not have bounding box information as a log file. Process the log file and create the missing bounding box information on the dataset. This method is suitable if you have a large quantity of data to process as you can split the report log file can be split into multiple files, allowing the utility to process fewer datasets in each execution:

```
create_or_update_bbox_and_tso -u=user> -p=password> -g=group>
-mode=query -translation_mode=JTT0BB0X+JTT0TS0
-output_dir=c:\temp
```

```
create_or_update_bbox_and_tso - u=user> -p=password> -g=group>
```

```
-mode=process -translation_mode=JTTOBBBOX+JTTOTSO
-dataset_list=c:\temp\file generated by above command>
```

- Query and process the datasets with a single command:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>
-mode=query+process -translation_mode=JTTOBBBOX+JTTOTSO
```

- Create bounding boxes for all NX datasets in the database. You can do this in one of two ways:
- Create a report file of the datasets that do not have bounding box information as a log file. Process the log file and create the missing bounding box information on the dataset. This method is suitable if you have a large quantity of data to process as you can split the report log file can be split into multiple files, allowing the utility to process fewer datasets in each execution:

```
create_or_update_bbox_and_tso -u=user> -p=password> -g=group>
-mode=query -translation_mode=NXBBOXTOBBBOX -output_dir=c:\temp
```

```
create_or_update_bbox_and_tso -u=user> -p=password> -g=group>
-mode=process -translation_mode=NXBBOXTOBBBOX
-dataset_list=c:\temp\file generated by above command>
```

- Query and process the datasets with a single command:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>
-mode=query+process -translation_mode=NXBBOXTOBBBOX
```

- Create a report file of all NX datasets in a log file that do not have an associated **UGPART-BBOX** form:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>
-mode=query -translation_mode=NXBBOXTOBBBOX
-output_dir=c:\temp\report.txt
```

- Delete all bounding box and TruShape data associated with a specified item revision:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>
-mode=delete -dataset=uid_of_item_rev1 or
absocc1[,uid_of_item_rev2 or absocc2, ...]
```

- Delete all bounding box and TruShape data associated with the parts specified in an input file:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>
-mode=delete -dataset_list=<path of file containing a list
of uids of item revisions or absolute occurrences separated
by new lines>
```

- Delete all bounding box data in the database:

```
create_or_update_bbox_and_tso -u=<user> -p=<password> -g=<group>  
-delete_all_bboxes
```

generate_tc_ps_path

Runs an ITK program that accepts the changed part list and generates the paths up to the top assembly for each item revision.

SYNTAX

```
generate_tc_ps_path [-u=user-id -p=password | -pf=password-file -g=group]
[-revision_rule=rule-name] -item_rev_list=rev-file-name
[-item_type=item-type] [-out_file=output-file-name]
-output_format= new | old [-configure_top_level_revs= yes | no]
-rev_full_file=rule-file-name [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item_rev_list

Defines the file containing the list of item revisions. Typically, this file is the output from the **find_released_item_rev** utility or the **find_recently_saved_item_rev** utility. This file can also be custom made.

-revision_rule

Defines the revision rule on the basis of which the configured parent is returned. These revision rules are used to determine whether they configure the item revisions specified in the **item_rev_list** file.

This argument may be repeated for various revisions rules.

This argument is not allowed if the **rev_rule_file** argument is defined, specifying the name of a file containing the list of revision rules. Otherwise, this argument is mandatory.

-out_file

Specifies the file to be used to write the utility output. If not defined, the output is written in the standard output.

-output_format

Specifies whether the utility output is in new format or old format. This argument is optional. If it is not defined, the new format is used. Possible values for this option are **new** and **old**.

The old format is as follows:

```
@DB/VEH0001/004,@DB/VPPS0002/001,@DB/VPPS0006/
001,@DBIA0009/002:Precise;Ap1p2 Best w/PDI
```

The new format is as follows:

```
PathPartRev@DBseparatorItem IDseparatorRevID/PartRev
[PartRev@DBseparatorItemIDseparatorRevID]/PartRev]
RevisionRuleRevisionrulename/RevisionRule/Path
```

-item_type

Specifies the item type of the top-level assembly. The utility lists only those paths with top-level assemblies of this type. In cases where such assemblies have parents, the defined path begins at the item with the given type. This argument is optional.

-rev_rule_file

Lists all the revisions rules to be used for the found item revisions. This argument is optional only if the revision rule arguments are defined; otherwise, this argument is required.

If both the **revision_rule** argument and the **rev_rule_file** argument are defined, the **rev_rule_file** argument takes precedence.

-configure_top_level_revs

Specifies whether to configure the top-level revision. This argument is optional.

If **Yes** is specified, the top-level item of the changed item revision is configured and the changed item revision is checked to determine if it uses the top-level item revision.

If **No** is specified, or the argument is not specified, the top levels are not configured.

-h

Displays help for this utility.

ENVIRONMENT

This utility should be run from a shell where the Teamcenter environment is set.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To generate a path for the item revisions specified in a text file:

```
generate_tc_ps_path -item_rev_list=released_items.txt  
-revision_rule=revision-rule-name -output_file=path_list.txt  
-output_format=new -configure_top_level_revs=no
```


identify_non_structure_edges

Marks parent and child items that are ignored when adding, listing, or removing entries in spatial indexes. There are four ways to use this utility.

- Adding and specifying what to add using parallel lists of item IDs or UIDs. For example, in a parallel list of parents and children, the first parent goes with the first child, the second parent goes with the second child, and so on.
- Adding and specifying what to add by a parent item type and child item type.
- Listing all elements or optionally specifying that only elements relevant to a particular product or particular parent or child objects be listed.
- Removing and specifying what to remove by giving parallel lists of item IDs or UIDs. For example, in a parallel list of parents and children, the first parent goes with the first child, the second parent goes with the second child, and so on.

SYNTAX

```
identify_non_structure_edges [-u=user-ID [-p=password | -pf=password-file]
[-g=group] ]
[-add {-parent_item_id={parent-object1, parent-object2, ... | @filename}
-child_item_id={child-object1, child-object2, ... | @filename} |
-parent_uid={parent-uid1, parent-uid2, ... | @filename}
-child_uid={child-uid1, child-uid2, ... | @filename} |
-parent_item_type=parent-type -child_item_type=child-type} ]
[-remove {-parent_item_id={parent-object1, parent-object2, ... | @filename}
-child_item_id={child-object1, child-object2, ... | @filename} |
-parent_uid={parent-uid1, parent-uid2, ... | @filename}
-child_uid={child-uid1, child-uid2, ... | @filename} } ]
[-list {-all [parent1, parent2, ... | @ filename] |
[child1, child2, ... | @ filename] }
{product={product1, product2, ...}
[-parent_item_id={parent1, parent2, ... | @filename}
[-child_item_id={child1, child2, ... | @filename} ] |
[-parent_uid={parent1, parent2, ... | @filename}
[-child_uid={child1, child2, ... | @filename} ] }
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-add

Adds entries for parent and child pairs. You can specify a parent and child object pair using item IDs or UIDs. You enter the identifiers as parallel, comma-separated lists for the parent and child flags.

For long lists of values, you can create parent and child object files that contain a separate identifier on each line. Use the *@file-name* values to supply the file names on the command line in place of the comma-separated lists.

Alternatively, you can use the **-parent_item_type=parent-type** and **-child_item_type=child-type** flags to specify a parent item and child item type pair.

-remove

Removes entries for parent and child pairs. You can specify a parent and child object pair using item IDs or UIDs. You enter the identifiers as parallel comma-separated lists for the parent and child flags.

For long lists of values, you can create parent and child object files that contain a separate identifier on each line. Use the *@file-name* values to supply the file names on the command line in place of the comma-separated lists.

-list

Lists all entries when used with the **-all** argument without the optional parent or child object specifications. You can filter the listed entries by specifying parent or child objects using item IDs or UIDs. You can narrow the filter by specifying both parent and child objects to get a list of entries containing the parent objects that have at least one of the specified child objects. You enter the identifiers as comma-separated lists for the parent and child arguments.

Optionally, you can list only those entries associated with one or more specified products using the **-product=product-name** argument. You can filter the product entries by specifying parent or child objects or both parent and child objects in the same way as the **-all** argument.

For long lists of values, you can create parent and child object files that contain a separate identifier on each line. Use **@file-name** values to supply the file names on the command line in place of the comma-separated lists.

-h

Displays help for this utility.

EXAMPLES

- To add parent-item and child-item pairs **pitem1**, **pitem2** and **citem1**, **citem2**, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group
-add -parent_item_id=pitem1,pitem2,pitem3
-child_item_id=citem1,citem2,citem3
```

- To add parent-item and child-item pairs **puid1**, **puid2** and **cuid1**, **cuid2**, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group
-add -parent_uid=puid1,puid2,puid3,puid4
-child_uid=cuid1,cuid2,cuid3,cuid4
```

- To add parent-item and child-item pairs where the parent item is of type **ItemType1** and the child item is of type **ItemType2**, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group
-add -parent_item_type=pitemtype
-child_item_type=citemtype
```

Note:

This example uses the parent and child item type, rather than the parent and child item ID.

- To list all parent-item and child-item pairs, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group -list
-all
```

- To list parent-item and child-item pairs by product, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group -list
-product=product1
```

- To remove parent-item and child-item pairs using item IDs, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group
-remove -parent_item_id=pitem1,pitem2,pitem3
-child_item_id=citem1,citem2,citem3
```

- To remove parent-item and child-item pairs using UIDs, enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group
-remove -parent_uid=puid1,puid2,puid3,puid4
-child_uid=cuid1,cuid2,cuid3,cuid4
```

- To remove parent-item and child-item pairs using a file containing the item IDs, create a parent and child file containing a single ID on each line. The pairs are determined by the order of the values in the file, for example:

Create a parentList.txt file with the following contents:

```
pitem1
pitem2
:
pitemN
```

Create the childList.txt file with the following contents:

```
citem1
citem2
:
citemN
```

Enter the following command on a single line:

```
identify_non_structure_edges -u=Tc-admin-user -p=password -g=group -remove
-parent_item_id=@parentList.txt
-child_item_id=@childList.txt
```

item_to_part_design

Converts instances of the **Item** class and its subtypes to specified instances of the **Part** and **Design** classes or their subclasses or subtypes. Use this utility if you have existing product structure containing items that you want to convert to separate part and design (CAD) objects. You must create a text file that provides the necessary input for the conversion process.

Note:

You must restart the Teamcenter server after running this utility for the new objects to be visible in the rich client.

The utility allows you to:

- Convert all objects in the system or a specified set of objects of a given source type to given target types.
- Specify the source as the **Item** class or its subtypes and specify the target as the **Part** and **Design** classes, their subclasses, or subtypes.

Caution:

After you run this utility, you cannot revert the changes.

You must analyze the impact on business rules (for example, property rules and GRM rules) of the conversion and then plan accordingly. The utility does not check the validity of the business rules.

Performance may be poor if you try to convert the entire database or a large structure. If possible, limit the structure size or the number of objects to convert.

The following conversions are supported by this utility:

- **Item** subtypes to **Part** subtypes
- **Item** subtypes to **Design** subtypes
- **Item** subtypes to **Part** class
- **Item** subtypes to **Part** subclass
- **Item** subtypes to **Design** class
- **Item** subtypes to **Design** subclass
- **Item** class to **Part** class
- **Item** class to **Part** subclass

- **Item** class to **Design** class
- **Item** class to **Design** subclass

Note:

Item subtypes are secondary business objects that have an **Item** class. Custom primary business objects that are created as a child of **Item** are not supported by this utility.

The utility operates in three modes:

- **type_based**

Input from the user is an item class or its subtype, which is a source type. The user specifies a target type. All objects in the database of source type are converted to the target type.

- **item_id_based**

In this mode, the user specifies a comma-separated list of item IDs, together with the source and the target types. Only the *valid* item IDs listed in the input file are converted from the source type to the target type.

- **structure_based**

The user specifies the item identifier of the top-level item in the structure, together with the source and the target types. This mode converts all the valid children of the top-level item to the target type.

The user must create a text file that contains input to the entire conversion process.

SYNTAX

```
item_to_part_design [-u=user-id -p=password | -pf=password-file -g=group]
{-mode=type_based | item_id_based [-itemid=item-id | -key_property_list=key-id-list]
| structure_based [-itemid=item-id | -itemkey=key-id] }
-file=input-file-name
[-rf=report-file-name] [-dryRun=true/on | false/off]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies **type_based**, **item_id_based**, or **structure_based**.

-itemid

Specifies the ID of the item to be exported. Valid only if no input file is specified using the **-i** option.

-itemkey

Specifies the key of the object. You can use the **-itemid** argument or the **-itemkey** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-key_property_list

Specifies the key attributes of the object except the item ID.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-file

This argument is mandatory for all the conversion modes.

Specifies the name of the input file containing conversion parameters, including the full path. The input file is a list of key-value pairs. The keys indicate the source and target item types. The input file has the following format:

```
source_class_name=Item class | SubType
target_class_name=Part/Design class | SubClass | SubType
item_id_list=Name of input file
```

Ensure you use internal names for the source and target.

Note:

The list of item IDs must end with a comma (,).

-rf

This optional argument specifies the name and location of the report file. If you do not specify a name and location, the system writes the file to *C:\temp* on Microsoft Windows systems or */tmp* on other systems. It is not applicable in **type_based** mode.

-dryRun

This optional argument causes the utility to validate if the specified items or structure can be validate, but does not perform the conversion. Valid values are **true** or **on** and **false** or **off** (case sensitive). It is not applicable in **type_based** mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

- Execute this utility when no other activity is present on the database.
- This utility changes only the type of item object, its revisions, master form, and revision master form.
- The storage class of source and target master forms is assumed to be same. If the attributes on the source and target form storage class are different, the utility does not know how to map those attributes. Hence the utility does not change the storage class of the form, but converts the form object itself to the target class. For example, while converting the standard **Item** class to **Design**, it performs the following conversion:

- Before conversion, the item master form (a form storage class) is an item master.
- After conversion, the design master form (a form storage class) is an item master.
- The utility converts objects in a particular database. If the objects being converted are exported to different sites, either the replica of the object must be deleted before conversion or replica objects also need to be converted in a similar fashion. If the replica objects are not converted due to change of type, synchronizing of data does not occur.
- If you use **item_id_based** mode or **structure_based** mode, items with export records or conflicting GRM rules are not converted. They are converted if you use **type_based** mode.
- The conversion of item subclass instances is not supported.
- The utility does not support source and target classes or subtypes other than those listed in the description.
- Any customization of the source object type is not supported after conversion to the target type.
- If the target is a subclass of **Part** or **Design** with custom attributes, those custom attributes of the converted instance or object are populated with default values, if any are defined. If none are defined, the custom attributes are null.

EXAMPLES

- The following example uses **type_based** mode and converts all the objects of a source type to a target type defined in the **Sample.txt** file:

```
item_to_part_design -u=Tc-admin-user -p=password -g=group
  -mode=type_based -file="C:\Sample.txt"
```

The contents of **Sample.txt** are:

```
source_class_name=Item
target_class_name=Part
```

- The following example uses **item_id_based** mode and converts all of the *valid* items listed in the **Sample.txt** file from source type to target type. No dry run is performed, but a report is written to the **ReportFile.txt** file.

```
item_to_part_design -u=Tc-admin-user -p=password -g=group
  -mode=item_id_based -rf="C:\temp\ReportFile.txt"
  -file="C:\Sample.txt" -dryRun=false
```

The contents of **Sample.txt** are:

```
source_class_name=Item
target_class_name=Part
item_id_list=000001,000004,000009,
```

- The following example uses **item_id_based** mode and converts all of the *valid* items listed in the **Sample.txt** file from source type to target type. A dry run is performed, and a report is written to the default location.

```
item_to_part_design -u=Tc-admin-user -p=password -g=group
-mode=item_id_based -file="C:\Sample.txt" -dryRun=true
```

The contents of **Sample.txt** are the same as in the previous example.

- The following example uses **structure_based** mode and converts all the *valid* items and children of the specified item identifier from a source type to a target type defined in the **Sample.txt** file. No dry run is performed, but a report is written to the **ReportFile.txt** file.

```
item_to_part_design -u=Tc-admin-user -p=password -g=group
-mode=structure_based
-rf="C:\temp\ReportFile.txt" -file="C:\Sample.txt"
-dryRun=false -itemId=000008
```

The contents of **Sample.txt** are:

```
source_class_name=Item
target_class_name=Part
```

Note:

item_id_list is not required.

- The following example uses **item_id_based** mode with the **key_property_list** argument. Because the multifield key is defined as **item_id,object_type**, only the **object_type** property is given (which is **Mfk2Item**).

```
item_to_part_design --u=Tc-admin-user -p=password -g=group
-mode=item_id_based
-key_property_list=object_type=Mfk2Item -file=itemlist.txt
```

multiple_svr_variant_configurator

Assists in creating configured structure representations using saved variant rules (SVRs) from unconfigured structure representations.

- Use the **bomwriter** utility to generate an unconfigured PLM XML file containing product structure information, including the variant conditions for child lines.
- Use the **multiple_svr_variant_configurator** utility to read the unconfigured PLM XML file and the specified SVRs.

The utility generates multiple pruned PLM XML files, corresponding to each SVR.

- Optionally, import pruned PLM XML files back into Teamcenter as **DirectModelAssembly** datasets using the **import_file** utility.
- Open the configured PLM XML files or datasets in Lifecycle Visualization.

Note:

Before running this utility, there must be a **DirectModelAssembly** dataset with a **TCEng_rdv_plmxml_unconfigured** relation placed under the product revision containing the unconfigured PLM XML file. Import the dataset using the **import_file** utility, ensuring that the variant XML preexists as a named reference of the unconfigured dataset.

SYNTAX

```
multiple_svr_variant_configurator [-u=user-id -p=password | -pf=password-file
-g=group] -product_id=product-item-ID -product_rev=product-revision-ID
-dataset_name=unconfigured-dataset [-log_file=file-name]
-directory_name=directory-name
{-svr_input_file=file-name | -process_all_svrs=Y} [-is_import_required=Y]
[-import_utility=path-to-utility] [-import_utility_parameters=utility-parameters] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter

database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-product_id

Specifies the product item ID. Specify the top level of the assembly for which you want to generate configured PLM XML files.

-product_rev

Specifies the product revision ID. Specify the top level of the assembly containing the SVRs.

-dataset_name

Specifies the name of the dataset under the product revision containing the unconfigured files. This must be a **DirectModelAssembly** dataset with a **TCEng_rdv_plmxml_unconfigured** relation placed under the product revision containing the unconfigured PLM XML file.

-log_file

Specifies the full path to the log file in which activity is recorded. Use this argument only when importing the pruned PLM XML files.

-directory_name

Specifies the full path to the directory in which the pruned PLM XML files are stored. The user running the utility must have write access to this directory.

-svr_input_file

Specifies the full path to the file containing the SVRs with which to configure the assembly.

You must specify either the **-svr_input_file** or the **-process_all_svrs** argument for the utility to run. If both are specified, **-svr_input_file** takes precedence.

-process_all_svrs

Specifies that all SVRs are processed. The valid value is **Y**.

You must specify either the **-svr_input_file** or the **-process_all_svrs** argument for the utility to run. If both are specified, **-svr_input_file** takes precedence.

-is_import_required

Specifies whether the pruned files are imported. Valid values are **Y** and **N**. The default setting is **N**.

If you set this argument to **Y**, you must set the **-import_utility** and **-import_utility_parameters** arguments.

-import_utility

Specifies the full path to the **import_file** utility, used to import the pruned files.

If you set this argument, you must set the **-import_utility_parameters** argument.

-import_utility_parameters

Sets the required parameters of the **import_file** utility. Specify parameters in a single string. Separate each parameter with a hash sign (#). For example:

```
-import_utility_parameters=#-u=userID#-p=password#-g=group>
#-d=ConfiguredAll#-ref=ConfiguredAssembly#-type=DirectModelAssembly
#-relation=TCEng_rdv_plmxml_configured#-desc=ConfiguredAll
#-use_existing=no#-f=#-item=ABC00004#-revision=001#
```

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Before running this utility, there must be a **DirectModelAssembly** dataset with a **TCEng_rdv_plmxml_unconfigured** relation placed under the product revision containing the unconfigured PLM XML file. Import the dataset using the **import_file** utility, ensuring that the variant XML preexists as a named reference of the unconfigured dataset.

EXAMPLES

- In the following example, all SVRs listed in the **input_svr.txt** file are used to generate multiple pruned PLM XML files, one file for each SVR. The files are not imported. The pruned PLM XML files are generated from the **multi_svr** directory.

```
multiple_svr_variant_configurator -u=userID -p=password -g=admin
-product_id=ABC004 -product_rev=001
-dataset_name=ABC00004Unconfigured2 -directory_name=
c:\temp\multi_svr -svr_input_file=c:\temp\multi_svr\input_svr.txt
```

- In the following example, all SVRs under the **ABC004/001** product revision are used to generate multiple pruned PLM XML files, one file for each SVR. The files are not imported. The pruned PLM XML files are generated from the **multi_svr** directory.

```
multiple_svr_variant_configurator -u=userID -p=password -g=admin
-product_id=ABC004 -product_rev=001
-dataset_name=ABC00004Unconfigured2
-directory_name=c:\temp\multi_svr -process_all_svrs=y
```

- In the following example, all SVRs under the **ABC004/001** product revision are used to generate multiple pruned PLM XML files, one file for each SVR. The files are then imported back into the Teamcenter database.

```
multiple_svr_variant_configurator -u=userID -p=password -g=admin
-product_id=ABC004 product_rev=001 -dataset_name=ABC00004Unconfigured2
-directory_name=c:\temp\multi_svr -process_all_svrs=y
-is_import_required=Y -log_file=c:\temp\multi_svr\log_file.txt
-import_utility={TC_ROOT}\bin\import_file
-import_utility_parameters=#-u=userID#-p=password#-g=admin
#-d=ConfiguredAll#-ref=ConfiguredAssembly
#-type=DirectModelAssembly#-relation=TCEng_rdv_plmxml_configured
#-desc=ConfiguredAll#-use_existing=no#-f=#-item=ABC00004
#-revision=001#
```

The parameters for the **import_file** utility are set in a single string, each parameter separated by a hash mark (#). The **-d** parameter specifies the prefix to the dataset names created upon import, followed by an underscore. For example:

ConfiguredAll_

You must include the **-f** parameter in the parameter string for the **import_file** utility. Do not assign it a value. The import utility automatically supplies the pruned PLM XML files.

ps_exportconfignxassembly

Enables a site to export a configured NX assembly. The assembly is configured using the given revision rule and saved variant rule.

SYNTAX

```
ps_exportconfignxassembly [-u=user-id -p=password | -pf=password-file -g=group]
-item=top-item-id | -key=keyAttr1=keyVal1 [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
-rev=top-rev-id [-revrule=revision-rule]
-variant=saved-variant-rule [-scopeitem=item-id]
[-scoperev=rev-id] [-display] [-verbose=y | yes | n |no]]
[-exclude=itemid1,itemid2,itemiid3,...
| -excludekeys=keyAttr1=keyVal1 [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]; keyAttr1=keyVal1
[,keyAttr2=keyVal2]...[,keyAttrN=keyValN];...] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a required argument unless the **TC_auto_login** site preference is set.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This is a required argument unless the **TC_auto_login** site preference is set.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

This is a required argument unless the **TC_auto_login** site preference is set.

If used without a value, the user's default group is assumed.

-item

Specifies the item ID of the item to be exported. This argument is required unless **-key** is defined. If both **-item** and **-key** are defined, **-key** takes precedence.

-key

Specifies the key of the item to be exported. This argument is required unless **-item** is defined. If both **-item** and **-key** are defined, **-key** takes precedence.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev

Specifies the revision ID of the top line of the structure to be exported. This argument is optional. If specified, the rest of the structure below the top line is configured by the **-revrule** argument. If not specified, the entire exported structure is configured by the **-revrule** argument.

-revrule

Specifies the revision rule to use to configure the structure. This argument is required.

-variant

Specifies the name of the variant rule to apply to configure the structure. This argument is required.

-scopeitem

This is an optional argument. Include this argument if and only if the given variant rule is not attached to the exporting item (top item) but is attached to this scope item's revision. The item revision is specified in the **scoperev** argument.

-scoperev

This is an optional argument. Include this argument if and only if the given variant rule is not attached to the exporting item (top item) but is attached to this **scoperev**.

-display

Displays the output folder to the screen. This argument is optional. If this argument is not specified, the output folder is not displayed at the end of the successful operation.

-verbose

Prints the debug statement. This argument is optional. The default value is **n**.

-exclude

Specifies the list of item IDs to exclude from exporting after the structure is configured using the revision rule and saved variant rule. This argument is optional.

-excludekeys

Specifies the list of keys to exclude from exporting after the structure is configured using the revision rule and saved variant rule. This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

NX must be installed and configured.

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

The files are created in the folder specified by the **TC_TMP_DIR** variable. The value of **TC_TMP_DIR** must be set to the desired destination directory temporarily before running the utility.

EXAMPLES

- Exports **TopAssmRevA** after configuring the structure below the top line with the **Latest Working** revision rule and **Tire200** saved variant rule. **SubAssm1** is excluded from the export even though it was configured.

```
ps_exportconfignxassembly -u=Tc-admin-user -p=password -g=group
  -item=TopAssm1 -rev =TopAssmRevA -revrule=Latest Working
  -variant=Tire200 -display -verbose=y -exclude=SubAssm1
```

- Exports **TopAssmRevA** after configuring the structure below the top line with the **Latest Working** revision rule and **Tire200** saved variant rule. There is no exclusion list provided in this example.

```
ps_exportconfignxassembly -u=Tc-admin-user -p=password -g=group
  -item=TopAssm1 -rev =TopAssmRevA -revrule=Latest Working
  -variant=Tire200 -display -verbose=n
```

- Exports **SubAssmRevA** after configuring the structure below the top line with the **Latest Working** revision rule and **Tire200** saved variant rule. The saved variant rule is attached to **TopAssmRevA**.

```
ps_exportconfignassembly -u=Tc-admin-user -p=password -g=group  
-item=SubAssm1 -rev =SubAssmRevA -revrule=Latest Working  
-variant=Tire200 -scopeitem=TopAssm1 -scoprev=TopAssmRevA
```

- Exports revision **A** of the item with key of **CarModel**. The structure below the top line is configured with the **Latest Working** revision rule and **Car1** saved variant rule.

```
ps_exportconfignassembly -u=xxx -p=yyy  
-key=item_id=CarModel -rev=A  
-revrule="Latest Working" -variant=Car1 -verbose=y
```

ps_rename_bvrs

Renames BOM views and BOM view revisions using the current naming scheme. The new name can differ from the old name because the naming scheme has changed or because the name of the view type has changed.

By default, the utility runs on all BOM views and BOM view revisions in the database, but it can accept an item ID argument (including wildcards) or a key argument, defining a set of objects to rename.

SYNTAX

```
ps_rename_bvrs[-u=user-id -p=password | -pf=password-file -g=group]
[-item=item_pattern]
| -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
[-view=view-type] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a required argument unless the **TC_auto_login** site preference is set.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This is a required argument unless the **TC_auto_login** site preference is set.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

This is a required argument unless the **TC_auto_login** site preference is set.

If used without a value, the user's default group is assumed.

-item

Specifies which BOM views or BOM view revisions to rename by a pattern match on the parent item ID.

-key

Specifies which BOM views or BOM view revisions to rename by key. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys in BMIDE for Data Model Design*.

-view

Specifies which BOM views or BOM view revisions to rename by BOM view type. The default is to rename BOM views or BOM view revisions of all types.

-v

Verbose mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the following preferences:

TC_ignore_case_on_search

TC_pattern_match_style

For more information about these preferences, see *Teamcenter Preferences*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Wildcard characters used in the **-item** argument may require enclosure in double quotation marks to prevent the shell from expanding them.

EXAMPLES

- To update names of all BOM views and BOM view revisions (BVRs) in the database, enter the following command:

```
$TC_ROOT/bin/ps_rename_bvrs
```

- Consider a site where the **Manufacturing** BOM view type is renamed to **M Site 1**. To rename all BOM views and BOM view revisions with parent item IDs beginning **pbx** to agree with the new name, enter the following command on a single line:

```
$TC_ROOT/bin/ps_rename_bvrs -view="M Site 1" -item="pbx*"
```

ps_traverse

Traverses a product structure and reports BOM line attributes and workspace attribute values in a file in delimited format. It also sets an assembly to precise and releases/transfers ownership of the item revisions that constitute the structure. The inputs for these are taken from a configuration file and the input for product structure configuration are taken from the command line options.

A configuration file for input is mandatory. If the **te.cfg** file exists in the current directory, it is considered. If not, a configuration file must be specified by the **-cfg** options. A sample configuration file is provide in the **\$TC_ROOT** directory.

SYNTAX

```
ps_traverse [-u=user-id -p=password | -pf=password-file -g=group]
-itemid=item-to-traverse
-rev=revision-for-item-to-traverse
[-revrule=revision-rule-to-configure-BOM-window]
[-viewtype=type-of-item-revision-BVR-to-traverse]
[-variant=saved-variant-object-to-configure-BOM-window]
[-log=log-file-for-session-output]
[-packlines= true | false]
[-cfg=configuration-file-for-options]
[-report=report-file-for-output]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-itemid

Specifies the ID of the item for which the associated BOM view revision (BVR) is traversed. (BOM view revisions are associated with revisions corresponding to the specified item.)

-rev

Specifies the revision of the item specified by the **-itemid** argument. The revision must have an associated BVR.

-viewtype

Specifies the type of the BVR to be traversed.

-revrule

Specifies the revision rule used to configure the BOM window. The default revision rule is **Latest Working**.

-variant

Specifies the saved variant used to configure the BOM window. If more than one saved variant exists, the first one found is considered.

-log

Specifies the log file to which the output is directed. The default file is **te.log**.

-report

Specifies the file to which the report is written. The default file is **tereport.txt**.

-packlines

Indicates whether to pack or unpack BOM lines.

-cfg

Specifies the input configuration file. The default file is **te.cfg**.

-h

Displays help for this utility.

CONFIGURATION FILE ENTRIES

The configuration file is a text file in which entries must be made under the following separate headings:

bomreport=

Valid values for this attribute are the display names of columns in Structure Manager, for example, **Rule configured by** and **Sequence No.**, and are case sensitive. Values listed in the columns are reported for each node in the BOM.

woreport=

The values of these object attributes in workspace listed under this entry are reported.

formattributes=

The form attributes to be reported are listed under this string. These should be in the format **Form Type Name:attribute**. The form values are truncated to 200 characters.

action=

The actions to be performed while traversing product structure are listed under this heading. The following actions are supported:

- **fastrelease**
- **changeowner**
- **setprecise**

Note:

When the **setprecise** action is specified, other actions and reporting inputs are ignored and the utility makes only the specified assembly precise.

relstat=

Specifies the release status to be applied if the specified action is **fastrelease**.

newowner=

Lists the new owner to whom the object ownership is transferred if the **changeowner** action is specified.

alternate=

Specifies whether alternates are processed. Valid values are **Yes** and **No**.

substitute=

Specifies whether substitutes are processed. Valid values are **Yes** and **No**.

delimiter=

Specifies the delimiter used to separate attribute values in report generation. The default delimiter is a semicolon (;).

columnwidth=

Specifies the attribute values used in report generation. The default column width is 20 characters.

EXAMPLES

Following is the content of the **ps_traverse.cfg** sample configuration file, which is located in the **\$TC_ROOT\sample\examples** directory.

```
alternate=
yes
bomreport=
BOM Line Name
woreport=
object_name
item_revision_id
owning_user
formattributes=
delimiter=
#
columnwidth=
25
action=
#fastrelease
#changeowner
#setprecise
relstat=
X
newowner=
tcdba
group=
dba
```

Sample configuration file

ps_upload

Creates an imprecise product structure based on an input file.

SYNTAX

```
ps_upload [-u=user-id -p=password | -pf=password-file -g=group] [-o= yes | no]
[-f] [-c= Item | Architecture] [-t=type] [-v] [-i=input-file] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-o

Specifies overwrite mode. Default value is **on**.

-f

Displays a help message about the input file format.

-c

Specifies the item class name to create. Specify either **Item** or **Architecture**.

-t

Specifies the default item type to create.

-v

Displays verbose information.

-i

Specifies the full path to an input file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

The default mode, **overwrite**, replaces any existing structures with the information contained in the input file.

INPUT FILE FORMAT

The input file consists of lines of comments, directives, and items. Each item and key line defines an item that the **ps_upload** utility creates. The comments and directive lines only alter the input file parsing behavior. They have no effect on the created items.

The structure hierarchy is determined by the level column (where the top level is level zero). The textual indentation of the item ID in the example file is only to make it more readable. Each time a level zero item is created, a new structure is started. Therefore, many structures (including single items) can be created from a single input file.

The top-level item in each structure is added to the user's **Newstuff** folder.

Note:

Do not show more than one occurrence of the same expanded assembly or you duplicate its contents.

The **ps_upload** utility assigns **A** as the initial revision ID.

With the exception of lines beginning with **#DELIMITER**, **#SUB_DELIMITER**, or **#COL**, the pound sign (#) in the first column indicates a comment that is ignored. Completely blank lines are also ignored.

#DELIMITER <i>x</i>	Specifies the delimiter. Default value is a space.
#SUB_DELIMITER <i>x</i>	Specifies the sub-delimiter. Default value is a semicolon ;.
#COL	Specifies the column heading order.

The format is **#COL** *field field field*

The default value is:

```
item arch_element_id name level seq occs
qty uom sub
```

key Specifies the key. To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

item Specifies the item ID.

rev Specifies a revision letter other than the default. The default is **A**.

Note:

You can create multiple revisions of the same item. Do this by first creating the item without specifying a level so that the item is not placed into any structure. Enter a line for each revision required. The **rev** column can then be left blank in the structure lines for items already created.

arch_element_id Specifies the architecture element ID.

option Specifies an option and set of allowed values to be defined and attached to the item being created by that line. The format is:

Option-name ; Value ; Value . . .

Note:

The delimiter must be that defined for substitutes (**#SUB_DELIMITER** directive). In the previous example, the delimiter is a semicolon (;).

loadif Specifies a simple variant condition using an option that has been defined in the previously created items. The variant condition is limited to one option/value expression. The format is:

Owning-ItemID;Option-name== or != Value

Note:

The delimiter must be that defined for substitutes (**#SUB_DELIMITER** directive). In the previous example, the delimiter was a semicolon (;).

name	Specifies the item name.
revname	Specifies the item revision name.
level	Specifies the structural hierarchy.
seq	Specifies the find number of the item in the BOM view.
occs	Specifies the number of occurrences of the item.
qty	Specifies the quantity of an item in the structure.
uom	Specifies the symbol representing the unit of measure.
sub	Specifies substitutes.
type	Specifies the item type. Note that the type must already exist in the database.
occname	Specifies the occurrence path name.
loadifkey	Specifies the key.

Substitutes

An entry in the substitutes column should consist of a delimiter-separated list of item IDs, where each substitute has been individually defined as a **level 0** structure. The default column delimiter is a semicolon (;).

The **#SUB_DELIMITER** directive tells the system that the next nonspace character on the line is used as a delimiter. If there is no nonspace character after the **#DELIMITER** directive, the delimiter is set to a blank space (' '). The substitute delimiter cannot be the same as the column delimiter.

EXAMPLES

The following figure illustrates the general file layout and shows the arbitrary use of the **#COL** and **#DELIMITER** directives:

```
# Example File for ps_upload
# The product structure for a bicycle.
# Change the delimiter to a comma, so we can use spaces in the name
#DELIMITER ,
# We start off with the default column order, and define a few simple
# parts that can be used as substitutes later. Note that these parts do
# not
# have a Level defined,
# and so will not be part of any structure.
#   item          name          Level Seq  Occs  Qty  Uom Sub
#   b100,         Bolt type 100
#   b101,         Bolt type 101
# Now we get on to the main structure
#   b001,         bicycle,       0,
#   b002,         frame,         1,  10
#   b003,         26" Wheel,     1,  20,
#   b004,         Metal Spoke,   2,  10,  20
#   b005,         bolt,         2,  20,  -,  2,  -,  b100 ;
b101
#   b003,         26" Wheel,     1,  30
# Change the column order to show how its done!
#COL   Level Seq item      Uom  Name                      Occs  Sub Qty
#   1,  40,  b007,      -,    Handlebars Assy,          2
#   2,  10,  b008,      -,    Brake Level Assy,          2
#   2,  20,  b009,      -,    Grips,                      2
#   2,  30,  b010,      -,    Handlebar Frame,
# Change Substitute Delimiter to /
#SUB_DELIMITER /
#   2,  40,  b005,      -,    bolt,                      -,  b101 / b100, 2
#   1,  50,  b011,      -,    Saddle,
# Change Delimiter to allow commas in the name
```

```
#DELIMITER $
      1$   60$  1001$    ml$  Oil, lubricating$   -$   -$   100
      1$   70$  1002$    m$   Paint, red$       -$   -$   2.4
-----
--
```

Product structure input file

purge_adhoc_configuration_contexts

Purges ad hoc configuration context objects older than the specified date and time to free space in the database. Such ad hoc configuration context objects are created by the **COLLABCTX_save_adhoc_configuration_context** ITK function as a temporary or intermediate storage location for stateless clients such as Active Workspace.

SYNTAX

```
purge_adhoc_configuration_contexts [-u=user-id -p=password | -pf=password-file -g=group]  
-date=DD-MMM-YYYY HH:MM:SS [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-date

Purges all ad hoc configuration contexts created before the date and time specified by this argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

None.

RESTRICTIONS

None.

purge_baselined_item_revisions

Purges baseline revisions when the automatic purge process fails.

SYNTAX

```
purge_baselined_item_revisions [-u=user-id -p=password | -pf=password-file
-g=group]
[-item_id=item_id
| -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
-status=release-status [-date=yyyy MM dd hh mm ss| now | today] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item_id

Specifies the appearance root item.

-key

Specifies the key of the appearance root item. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-status

Purges baselined item revisions of the status specified by this argument.

-date

Purges baselined item revisions created before the date/time specified by this argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

None.

RESTRICTIONS

None.

qsearch_process_queue

Updates or queries the spatial indexes used by the cacheless search mechanism. You can also use this utility to modify or query the state of the update queue process that updates these indexes.

When using this utility, the term *product* refers to the item at the top of a structure. If you make the top-level item indexable, the entire structure is indexed. Conversely, the term *item* or *object* refers to any item at any level of the structure, for example, one that represents a subassembly.

You can specify a product ID in the format **-product=product** or **-product_uid=product**. Specify an object ID in a similar format.

SYNTAX

```
qsearch_process_queue [-u=user-id -p=password | -pf=password-file -g=group]
[-item_id=item-id-pattern1,item-id-pattern2,item-id-pattern3,... | @filename]
{[-list_queue | -list_all_queue]} [-show_queue_oldest_date]
[-process_queue] [-process_queue_repeatedly [-delay=N] [-repeat=M]]
[-clear_queue] [-force_queue_update Objects]
[-force_queue_substructure_update=Objects]
[-force_queue_all_leaf_item_updates] [-force_queue_all_possible_updates]
[-force_queue_all_necessary_updates] [-force_queue_all_inconsistent_updates]
[-tolerance=Percentage]] [-ask_global_search_box_delta] [-list_volumes=Objects]
[-list_index_boxes=Objects] [-list_structure_index_boxes=Objects]
[-list_all_index_boxes]
[-clear_indexes] [-clear_structure_indexes=Objects] [-clear_all_indexes]
[-clear_queue_processed] [-clear_all_queue]
[-check_indexes=Objects]
[-check_structure_indexes=Objects]
[-list_suggested_updates[=filename] | -force_queue_suggested_updates |
-follow_only_check_failures [-find_cycles]
[-count_occurrencesobjects]
[-count_substructureobjects]
[-list_legacy_transforms [products]]
[-generate_dot_info objects [-nofollow] [-include_indexes]
[-include_bboxes] [-include_transforms][-outfile= file]]
[-find_datasets_below objects]
[-enable_product_scoping]
[-disable_product_scoping [-drop_tables]]
[-list_products]
[-make_indexable [products]]
[-make_non_indexable [-silent ] {products | objects | -all }]
[-list_indexable {products | objects | -all } [-additional_types=occthread,apn] [-outfile=file]]
[-check_indexable {products | objects} [ -below]]
[-clear_product_indexes products]
[-check_product_indexes [products]]
[-dump_substructure]
[-task=task-list] [-verbose] [-print_names] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item_id

Specifies a list of item ID patterns to process when you use product-scoped searches. Entries must be separated with commas. You can also supply the item ID patterns in a separate text file, with one item ID pattern per line. In this case, use the file name instead of the list of item ID patterns, preceding it with @. If you do not specify item IDs, all items in the database are processed.

-list_queue

Lists all unprocessed entries in the queue.

-list_all_queue

Lists all entries in the queue, including processed entries.

-show_queue_oldest_date

Shows the creation date of the oldest unprocessed entry in the queue.

-process_queue

Processes the unprocessed entries in the queue.

-process_queue_repeatedly

Processes the unprocessed entries in the queue repeatedly. Optionally, you can wait N seconds between process runs (default delay is 5 seconds). You can also specify a maximum of M times for the process to repeat (default is forever).

-delay

Specifies a delay of N seconds between process runs.

-repeat

Specifies a maximum of M times for the process to repeat (default is forever).

-clear_queue

Clears the unprocessed entries from the queue.

-force_queue_update

Adds an entry to the queue for the specified objects.

-force_queue_substructure_update

Adds an entry to the queue for the leaf items of the assembly beneath the specified objects. This action updates the entire assembly.

-force_queue_all_leaf_item_updates

Adds an entry to the queue for all leaf items. This action updates all assemblies.

-force_queue_all_possible_updates

Adds an entry to the queue for the primary of each **TC_bounding_box** relation.

-force_queue_all_necessary_updates

Adds an entry to the queue for the primary of each **TC_bounding_box** relation that lacks an index.

-force_queue_all_inconsistent_updates

Adds an entry to the queue for the primary of each **TC_bounding_box** relation with an apparently inconsistent index. You can optionally specify the percentage inconsistency to ignore, overriding the default value of 10%.

-tolerance

Specifies the percentage of inconsistency to ignore.

-ask_global_search_box_delta

Calculates the current global search box delta.

-list_volumes

For each specified object, lists the total volume occupied by all its contributing bounding-boxes and the total volume of all its index-boxes.

-list_index_boxes

Lists the index boxes for the specified objects.

-list_structure_index_boxes

Lists the index boxes for all possible configured structures for the given objects.

-list_all_index_boxes

Lists the index boxes for all objects.

-clear_indexes

Removes the indexes from the specified objects.

-clear_structure_indexes

Removes the indexes for all possible configured structures from the specified objects.

-clear_all_indexes

Removes the indexes from all objects.

-clear_queue_processed

Clears the processed entries from the queue.

-clear_all_queue

Clears all entries from the queue, that is, both processed and unprocessed entries.

-check_indexes

Checks the indexes for the specified objects.

-check_structure_indexes

Checks the indexes for all possible configured structures for the specified objects.

- If you specify **-list_suggested_updates**, the utility lists the objects for should be updated to fix any incorrect indexes detected for the structure. The list is written to the specified file or **stdout** if no file is specified. The file is written in a format suitable for **qsearch_process_queue -force_queue_update -uid=@filename**.
- If you specify **-force_queue_suggested_updates**, the utility adds entries to the queue to fix any incorrect indexes detected for the structure.
- If you specify **-follow_only_check_failures**, the utility assumes that if the indexes of the specified object are correct, the indexes for the entire substructure are also correct.

-find_cycles

Finds all cyclical structures.

-count_occurrences

Counts all occurrences in all structures of all items. Optionally, you can just count all occurrences of specified objects.

-count_substructure

Counts the substructure of all root items. Optionally, you can just count the substructure of specified objects.

-list_legacy_transforms

Lists all legacy transforms for all items in the database. Optionally, you can specify one or more products and the utility lists only legacy transforms for items in the specified products.

-generate_dot_info

Generates information about one or more specified objects and their ancestors in dot format suitable for post-processing to show cycles. Optionally, you can specify **-nofollow** to include only information about the specified objects' immediate ancestors. If you use the output dot file for post-processing to detect cycles, ensure you specify all items you expect to be part of the cycles in the *objects* argument, otherwise do not use this option.

You can optionally include information about the indexes, bounding-boxes, and transforms in the dot file.

You can optionally specify the name and location of the output file, otherwise the information is written to **stderr**.

-find_datasets_below

Return all datasets below one or more specified objects.

-enable_product_scoping

Limits searches to a specified product.

-disable_product_scoping

Disables the limiting of searches to a specified product. Optionally, you can also drop the indexability tables associated with the product scoped search.

-list_products

Lists all products available for scoped searches.

-make_indexable

Populates the indexability tables with data necessary to make the specified product or products searchable.

-make_non_indexable

Removes data associated with the specified products or objects from the indexability table, so making them unsearchable. You can also make all products and objects unsearchable. By default, you are prompted for confirmation but if you specify the **-silent** argument, no confirmation is necessary.

-list_indexable

Lists the contents of the indexability tables associated with the specified products or objects. You can also list the contents for all products and objects. By default, all indexable item IDs are listed. Optionally, you can generate a comma-separated list of occurrence threads, appearance path nodes

(APNs), or both. By default, the list is written to **stderr** but you can optionally specify the name and location of an alternate output file.

-check_indexable

Checks the indexability tables contain all the data necessary to index the specified products or objects. Optionally, you can specify the **-below** argument to check everything below a specified object, rather than just the object.

-clear_product_indexes

Remove the indexes from the specified product or products.

-check_product_indexes

Checks the indexes for the specified product or products.

-dump_substructure

Displays a list of item UIDs under the item ID specified by the **-item_id** argument.

-task

Specifies a task list of multiple arguments. Omit the leading dashes and separate entries with commas. For example:

```
-task=list_queue,process_queue
```

-verbose

Runs the utility in verbose mode to display the maximum amount of information. Typically, nonverbose utility sessions only display error messages.

-print_names

Prints item IDs and names as well as UIDs.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Where appropriate, objects may be specified as a list of UIDs or a list of item ID patterns, as follows:

- A list of UIDs must be preceded by **-uid=** and entries separated with commas:

```
-uid=uid1,uid2,uid3,...
```

You can also supply the UIDs in a separate text file, with one UID per line. In this case, use the file name instead of the list of UIDs, preceding it with **@**:

```
-uid=@filename
```

- A list of item ID patterns must be preceded by **-item_id=** and entries separated with commas:

```
-item_id=item-id-pattern1,item-id-pattern2,item-id-pattern3,...
```

You can also supply the item ID patterns in a separate text file, with one item ID pattern per line. In this case, use the file name instead of the list of item ID patterns, preceding it with **@**:

```
-item_id=@filename
```

EXAMPLES

The following example lists all unprocessed entries in the queue:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group -list_queue
```

The following example adds an entry to the queue for each of the objects listed in the **c:\temp\objfile.txt** file:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group  
-force_queue_update -uid=@c:\temp\objfile.txt
```

The following example checks the indexes for all objects that have an item ID prefixed with **123**:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group  
-check_indexes -item_id=123*
```

The following example lists the unprocessed entries in the queue and then processes them:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group  
-task=list_queue,process_queue
```

RESOLVING CIRCULAR REFERENCES

Some assemblies may contain circular references (cycles). If cycles exist in an assembly, you may encounter poor performance when creating indexes and running searches. If so, use this utility with the **find_cycles** argument to identify all such cycles. You should then remove the cycles manually before

executing other cacheless search utilities. You should also identify why these cycles are created and establish a process to prevent the creation of more cycles.

You can run the utility with the **find_cycles** argument against a single item or the entire database, as shown in the following examples.

Processing the entire database:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group -find_cycles 2>
c:\temp\list_of_cycles.txt
```

Processing a single item:

```
qsearch_process_queue -u=Tc-admin-user -p=password -g=group -find_cycles
-item_id=AKT75443
```

Note:

The utility output is written to standard error and you should redirect it to a file. For example, specify `2< c:\temp\list_of_cycles.txt`, as shown in the first example.

The output format is similar to the following example:

```
Cycle begins =====
Y31VRf_fQyYmUC AKY43846-LV2.X 1C PROFILE SECTIONS
YW7VRf_fQyYmUC AKY43841-LV2.X 1C Y_SECTIONAL VIEW
i70lXf0fgzknZC ATR04589-T04A WS LWR AT CL
h38NawpPQyYmUC AKT75443-TOTAL VEHICLE
i01NNpCxQyYmUC AKU25433-VEHICLE INTEGRATION MODULE
ii2NNpCxQyYmUC AKU25441-COMPARTMENT VEHICLE
yP2Vwa5kQyYmUC AKY08647-LV2.X HYB CDH 1B GA
XFWVwmjAQyYmUC AKY10317-LV2.X CDH STRUCTURE REF ASM
LZ8Vw$FeQyYmUC AKY27566-LV2.X HYB CDH MIKE FRAEYMAN PART
iiMl7dfigzknZC ATR05553-IA-CDH-REAR COMPT-MM
VH1VB_7gQyYmUC AKY41691-CDH RR RAIL U/B SI
aC6VRf_fQyYmUC AKY43866-LV2.X HYB CDH 1C GA

Cycle ends =====
```

To report cycles found when you execute an attribute search in Structure Manager or Design Context, set the **QS_START_CIRCULAR_STRUCTURE_CHECK_SIZE** and **QSEARCH_DEBUG_TEXT** user preferences to 1.

The output report format in this mode is as follows:

```
>>>> Found cyclic structure.UnconfiguredStep: STUDY MODULE ( 68342) - 0 -- 0 - 76915
(VPPS_20 STUDIES CHASSIS) ( zMMNRX46gzknZC) -- 77314 (E1JNRX46gzknZC) -- 0 --- 75941
(STUDY-GMT172/7 SPARE TIRE) ( SQ$Jb7D_QyYmUC)-- 76845 ( zc6J7309QyYmUC) -- 0 --- 73558
```

```
(TOTAL VEHICLE) (xTw01m0JgzknZC) -- 75523 ( jh3Jd9saQyYmUC) -- 0 --- 70375
(STUDY MODULE) ( RXw01m0JgzknZC) -- 71949 ( RuEFkrRSgzknZC) -- 0 --- 68342
(STUDY MODULE) ( TEHNRX46gzknZC) -- 69364 ( U10NRX46gzknZC) -- 0 --- 67615
(VPPS_40 STUDIES INTERIOR) ( zkMNRX46gzknZC) -- 67831 (ExNNRX46gzknZC) -- 0 --- 65940
(STUDY GMT172/177 FD SPEAKER PKG) ( THyJbqzAQyYmUC) -- 66630 (DVzNwd24QyYmUC) -- 0 --- 63228
(IA-FRT S/D HARDWARE LH 1&2LA00) (jEDBpNkogzknZC) -- 64916 ( lt4JbqzAQyYmUC) -- 0 --- 61430
(BOLT) ( wT1FNGA$QyYmUC) -- 62442 ( RdHFGSL$gzknZC) -- 0 ---
```

To clean up the cycles, search for the associated item in My Teamcenter. Do not perform a where used search or browse in Structure Manager because the applied revision rule prevents you from finding the items. When you find the item, send the associated BVR to Structure Manager and manually remove the cycle. In the previous example, look in the My Teamcenter **View** folder for every revision of item **AKY43866-LV2.X HYB CDH 1C GA** and remove the cycles in Structure Manager as you find them.

rdv_context_download

Evaluates or reevaluates a structure context object (SCO) and writes Design Context information and user attributes into a PLM XML or AJT (**BOMWriter**) file. The contents of the SCO can represent the entire product or a subassembly. The information written into the PLM XML file depends on the specified input transfer mode.

The output file is used to facilitate the sharing of data between users, programs, or sites using Multi-Site Collaboration. You can download this file to the operating system or import it into Teamcenter as a dataset.

This utility can process a single SCO, a list of SCOs, or all SCOs in a specified folder. If you process a list or folder of SCOs, a separate PLM XML file is generated for each SCO. It can also accept a saved query as input.

If you use Multi-Site Collaboration, the **validate_and_replicate_assembly** utility can consume the PLM XML files created by this utility and replicate them across sites.

If the output PLM XML file is in **BOMWriter** format, you can launch it in Lifecycle Visualization.

SYNTAX

rdv_context_download

```
[-item_id=item-ID] | [-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
[-rev_id=revision-ID] [-variant_rule_name=variant-rule-name]
[-revision_rule_name=revision-rule-name] [-engg_change_id=engineering-change-ID |
-engg_change-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
[-folder_name=folder-name] [-process_name=process-name]
[-zone_name=zone-name] [-zone_type=BOX | PLANE]
[-operator=BOX(Within | Outside | Interferes) | Plane(Above | Below | Intersects)]
[-sco_download=yes | true | YES | TRUE]
[-user_attribute=user-attribute]
[-sco_name(s)=StructureContextObjectName(s) | -saved_query_name=Teamcenter-saved-query-
name | -sco_folder=Teamcenter-folder-name] [-transfer_mode=transfer-mode-name] [-
output_format=BOM_writer FormatAJT | PLMXML | PIEPIE-PLMXML] [-file-name=file-name-without-
extension] [-absolute_path=path-of-file-to-be-stored-without-extension] [-h]
```

ARGUMENTS

-item_id

Specifies the item ID.

-key

Specifies the key of the item. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev_id

Specifies the revision ID.

-variant_rule_name

Specifies the name of the variant rule configuring the structure.

-revision_rule_name

Specifies the name of the revision rule configuring the structure, which defaults according to the **TC_config_rule_name** preference.

-engg_change_id

Specifies the engineering change item ID. The utility configures an RDV context based on change attachments and the latest engineering change revision.

-engg_change_key

Specifies the key of the engineering change item. The utility configures an RDV context based on change attachments and the latest engineering change revision.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-folder_name

Configures a context based on attachments of the *folder/envelope/engineering-change-revision-name*.

The attachments include the following:

- One product item revision
- One or more component item IDs
- Optional revision rule, overwrites the **-r** argument
- Optional variant rule, overwrites the **-v** argument

Note:

Search results are affected by the following preferences:

- **TC_config_rule_name**

- **WebDesignContextDefaultSearchDistance**
- **PortalDesignContextMaxMatchingObjects**
- **PortalDesignContextMaxMatchingBOMLines**

For more information, see the *Teamcenter Environment Variables*.

-process_name

Specifies the name of the workflow processes that have not yet completed.

-zone_name

Specifies the name of the zone. When both the **-zone_name** and **-zone_type** arguments are specified, the utility performs a search according to the preference settings.

-zone_type

Specifies the type of the zone, either **BOX** or **PLANE**. The **-zone_name** argument must be used in conjunction with the **-zone_type** argument.

-operator

Specifies the zone type operator. Valid values for the **BOX** zone type are **Within**, **Outside**, or **Interferes**; the default value is **Within**. Valid values for the **PLANE** zone type are **Above**, **Below**, and **Intersects**; the default value is **Intersects**. The **-zone_name** and **-zone_type** arguments must be specified in conjunction with the **-operator** argument.

Note:

This utility performs a proximity search if the **-zone_name** argument is not specified and performs a name zone search if the **-zone_type** argument is not specified.

-sco_download

Specifies if the SCO should be downloaded.

-user_attribute

Specify user attributes in the same format described for the **bomwriter** utility.

-sco_name

Specifies the name of the structure context object from which a PLM XML is generated. You can specify more than one SCO as a comma-separated list, for example:

SCO1,SCO2,SCO3

-saved_query_name

Specifies the name of a Teamcenter saved query that retrieves a set of SCOs to process for given search criteria.

-sco_folder

Specifies the name of a Teamcenter folder that contains a set of SCOs to process.

-transfer_mode

Specifies the name of a transfer mode to generate the PLM XML file. If no mode is specified, the SCOs are evaluated with the default transfer mode.

-output_format

Specifies the output file format, either BOM writer PLM XML or AJT, or PIE PLM XML. The default format is BOM writer PLM XML.

-file_name

Specifies the name of the output file to which the data is output. Provide the file name without an extension.

-absolute_path

Specifies the full path of the AJT or PLM XML file where the data is to be stored. If not specified, the utility looks at the value of the **RDVContextDownloadDirectory** preference. Otherwise, the current working directory is used as the default path.

Note:

The **absolute_path** must be specified without an extension. For example, **/users/x_user/tempfile** for Linux or **c:\temp\tempfile** for Windows.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the **RDV_debug** environment variable. If this variable is set, the Teamcenter **syslog** file contains additional debugging information.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

You must have the necessary create or modify privileges to successfully run this utility. If you do not have the appropriate privileges, data is not imported and an error message is written to the log file.

EXAMPLES

- The following example creates a PLM XML file called **TL109375** and uses the **TC_config_rule_name** preference to determine the revision rule:

- The following example prompts the user for autologin and creates an ASCII JT assembly file named **assy_ajt_file**:

```
$TC_ROOT/bin/rdv_context_download -folder=assy_folder  
-filename=assy_ajt_file -bom_writer_format=AJT
```

The following code sample shows the sample AJT file created in this example.

[illegible]

Sample AJT file

rdv_migrate_architecture

Effective from Teamcenter 2007.1 MP7, Platform Designer allows you to revise architecture breakdowns and carry forward all architecture breakdown elements (ABEs), variability, named variant expressions (NVEs) and part solutions to the next revision. Architectures created in previous versions of Teamcenter are not compatible with the current format and must be updated.

Use this utility to update all existing ABEs to the new format. After migration, each ABE is associated with two sets of appearance path nodes (APNs) and each APN is linked to an **AbsOccData** object. One **AbsOccData** object has its immediate parent as the context and the other **AbsOccData** object has the top-level architecture of the architecture breakdown as its context. The suppression flag on the first **AbsOccData** object is set to **true**, and the second suppression flag is set to **false**.

Note:

Use the **rdv_migrate_part_solutions** utility to migrate the associated part solutions.

SYNTAX

```
rdv_migrate_architecture [-u=user-id {-p=password | -pf=password-file} -g=group]
{-arch_ItemId<TOP-ARCH-ID> |
[-arch_key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]}
-arch_RevId<TOP-ARCH-REVISION-ID>
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-arch_ItemId

Specifies the item ID of the top architecture element.

-arch_key

Specifies the keys of the item to track. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the **RDV_debug** environment variable. If this variable is set, the **syslog** file contains additional debugging information.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Run this utility with a user ID that has sufficient permissions to create tasks and to write to the specified architecture breakdown and ABE objects in the database.

EXAMPLES

- The following example migrates a top architecture element with an item ID of **architecture001**:

```
rdv_migrate_architecture -u=Tc-admin-user -p=password -g=group
-arch_ItemId=architecture001
```

rdv_migrate_part_solutions

Effective from Teamcenter 2007.1 MP7, Platform Designer allows you to revise architecture breakdowns and carry forward all architecture breakdown elements (ABEs), variability, named variant expressions (NVEs) and part solutions to the next revision. Part solutions created in previous versions of Teamcenter are not compatible with the current format and must be updated.

Use this utility to migrate existing part solution data to the current format. The utility takes the top architecture element of an existing breakdown and fetches all the lines of usage for each of the ABEs in that breakdown. It then creates a **LOUHOLDER** holder under the top line, if one does not already exist. It adds all the part solutions under this holder and associates them with the corresponding ABEs with **Appearance Group** relations.

Note:

Use the **rdv_migrate_architecture** utility to migrate the associated architecture.

SYNTAX

```
rdv_migrate_part_solutions [-u=user-id {-p=password | -pf=password-file} -g=group]
{-arch_ItemId<TOP-ARCH-ID> |
[-arch_key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]}
-arch_RevId=<TOP-ARCH-REVISION-ID>
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-arch_ItemId

Specifies the item ID of the top architecture element.

-arch_key

Specifies the architecture breakdown keys of the item to track. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

-arch_RevId

Specifies the revision identifier of the same top architecture element.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the **RDV_debug** environment variable. If this variable is set, the **syslog** file contains additional debugging information.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Run this utility with a user ID that has sufficient permissions to create tasks and to write to the specified objects in the database.

EXAMPLES

- The following example migrates the part solutions in a top architecture element with an item ID of **architecture001** and revision identifier of **architecture001Rev1**:

```
rdv_migrate_part_solutions -u=Tc-admin-user -p=password -g=group
  -arch_ItemId=architecture001 -arch_RevId=architecture001Rev1
```

rdv_run_audit_report

Generates audit reports that report the information related to mismatches that exist between VAS CAD structure and usage data that is imported into Teamcenter.

Audit reports allow you to check that design solutions exist for each option combination (NVE) required on the architecture elements. This feature assumes you use both part and design solutions; if you use only part solutions, do not use this report.

Note:

You create audit reports for design solutions in Structure Manager. Use Platform Designer to create audit reports for part solutions.

To create an audit report for design solutions, Teamcenter first performs a consistency check, verifying that for each design solution in the selected installation assembly, there is a part solution that matches the NVE on the design solution. It also checks that the variant condition on the design solution is consistent with the variant condition on the part solution. If the NVE changes on the architecture or part solution, the variant condition on the design solution may be out-of-date and require refreshing. Alternatively, the design solution itself may be changed to meet the new NVE requirement.

Designers perform this type of audit on the installation assembly for which they are responsible. You cannot perform this audit on a level higher than the installation assembly.

You can still use the audit report if no part solutions exist yet, because the NVEs on the architecture are checked. You can create an audit report for a particular configuration by setting the variant and revision rules audit algorithm details. The audit checks the following:

- If the NVE is not referenced by any solution.
- If the NVE is referenced by one or more solutions with a matching architecture element ID, but the variant condition is out-of-date (assuming there are no split NVEs).
- If the usage quantity is not a positive integer.
- If the solutions are referencing an NVE, but the architecture element ID references a split NVE.
- If the NVE is referenced by one or more solutions, but none of them have an architecture element ID that matches that of the NVE. The audit report delivers a list of architecture breakdowns in rows, with colored indicators specifying whether the line is an exact, partial, or mismatch with respect to an NVE, part number (typically the part solution ID), and usage quantity.

The audit algorithms use data stored in occurrence notes on the design solution. It is therefore necessary to replace a design in the product using the Replace Design in Product wizard before the audit can be run to populate the appropriate occurrence notes. These occurrence notes are:

Usage_Product

Usage_PartNumber
Usage_Quantity

Configure these occurrence notes by setting the **RDV_copied_occurrence_notes** preference.

SYNTAX

```
rdv_run_audit_report [-u=user-id {-p=password | -pf=password-file} -g=group]
[-product_id=item-id-of-top-level-product | -product_key=key-id-of-top-level-product]
-product_rev_id=revision-id
-v_rule=saved-variant-rule
-rev_rule=revision-rule-name
[-input_file=input-file-path | -keys_input_file=keys-input-file-path]
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-product_id

Specifies the item ID of the top-level product.

-product_key

Specifies the key of the top-level product. This argument can be used to identify an item instead of the **-product_id** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-product_rev_id

Specifies the item ID of the top-level product revision.

-v_rule

Specifies the saved variant rule.

-rev_rule

Specifies a named revision rule. Defaults to the site default, frequently the **latest working** revision.

-input_file

Specifies the path to the file used for input when you use the **-product_id** argument.

-keys_input_file

Specifies the path to the file used for input when you use the **-product_key** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

rdv_set_default_variant_condition

Checks whether the input item has any qualified occurrences attached to it and sets a default variant condition on them so that the occurrence can participate in variant configuration. This utility configures the structure using the revision rule provided as input. If the revision rule is not specified, it uses the default revision rule.

This utility also sets the variant condition specified as input. If the variant condition is not specified then it uses the variant condition specified in the **RDV_default_variant_condition** preference.

SYNTAX

```
rdv_set_default_variant_condition [-u=user-id {-p=password | -pf=password-file}
-g=group]
[-item=item-ID | -key=key-ID]
[-revision_rule=revision-rule-name]
-variant_condition=condition-name
-delimiter=delimiter-string
-qualified_occu_scan_depth=depth
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the ID of the item to be used for the variant condition.

-key

Specifies a string used to identify an item instead of specifying an item using the **-item** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-revision_rule

Specifies a named revision rule. Defaults to the site default, frequently the **latest working** revision.

-variant_condition (Optional)

Specifies the variant condition.

-delimiter (Optional)

Specifies the delimiter to place between records.

-qualified_occu_scan_depth (Optional)

Specifies the level to scan occurrences.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following example configures the structure using default revision rule, scans the item for null variant components, and sets the default variant condition specified in preference files:

```
rdv_set_default_variant_condition -item_id=TL109375
```

- The following example configures the structure using the latest working revision rule, scans the item for null variant components, and sets the default variant condition specified in preference files:

```
rdv_set_default_variant_condition -item_id=TL109375  
-revision_rule=Latest Working
```

- The following example configures the structure using the latest working revision rule, scans the item for null variant components, and sets the variant data (**Color=Red**) on null variant components:

```
rdv_set_default_variant_condition -item_id=TL109375  
-revision_rule=Latest Working -variant_condition=002763:Color:Red
```

start_sco_dispatcher

Starts the structure context object (SCO) dispatcher to initiate the transfer of SCOs created by the **rdv_context_download** utility to other sites. The **validate_and_replicate_assembly** utility subsequently consumes the PLM XML files created by the **rdv_context_download** utility and replicates them across sites.

SYNTAX

```
start_sco_dispatcher [-u=user-id {-p=password | -pf=password-file} -g=group]
-saved_query_name=Teamcenter-saved-query-name |
-sco_folder=Teamcenter-folder-name |
-sco_list=list-of-scots | -sco_name=StructureContextObjectName
-site_name=Destination-site-name | -site_url=Destination-site-url
[-transfer_mode=transfer-mode-name] [-dataset_time_interval=time-interval]
[-user_attrs=user-attributes] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-sco_name

Specifies the name of the SCO from which a PLM XML file is generated. You can specify more than one SCO in a comma-separated list, for example:

SCO1,SCO2,SCO3

-sco_list

Specifies a list of SCOs to process. You can specify more than one SCO in a hash-separated list, for example:

SCO1#SCO2#SCO3

-saved_query_name

Specifies the name of a Teamcenter saved query that retrieves a set of SCOs to process for given search criteria.

-sco_folder

Specifies the name of a Teamcenter folder that contains a set of SCOs to process.

-transfer_mode

Specifies the name of a transfer mode to generate the PLM XML file. If no mode is specified, the SCOs are evaluated with the default transfer mode.

-site_name

Specifies the names of the sites with databases to which Teamcenter sends the SCOs. You can specify more than one site in a hash-separated list, for example:

Site1#Site2#Site3

-site_url

Specifies the URLs of the sites to which Teamcenter sends the SCOs. Use URLs, rather than site names, if the target sites do not have databases. You can specify more than one URL in a hash-separated list, for example:

Site1#Site2#Site3

-dataset_time_interval

Specifies the interval in minutes at which the **rdvcontextdownload** translator generates PLM XML files. If no interval is specified, the default value of 60 minutes is used.

-user_attribute

Specify user attributes to include in the PLM XML file. The specified attributes must be BOM line properties. You can specify more than one user attribute as a hash-separated list, for example:

Owningsitename#ItemID#RevisionID##last_mod_date

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- You must have the necessary create or modify data privileges to successfully run this utility. If you do not have the appropriate privileges, data is not imported and an error message is written to the log file.
- If the utility evaluates an SCO and identifies that the target BOM lines are not valid, it exits and displays an error message to the user. It does not process any further SCOs.

EXAMPLES

- The following example runs a saved query called **SCO_SQR** to transfer SCOs to a single database site using Multi-Site Collaboration:

```
start_sco_dispatcher -saved_query_name=SCO_SQR -site_name=testsite  
-dataset_time_interval=120
```

- The following example transfers all SCOs in the **SCO_folder** folder to a single site without a database. The transfer is achieved via FMS by specifying the target site URL:

```
start_sco_dispatcher -folder_name=SCO_folder  
-transfer_mode=PIEPLMXMLDEFAULT  
-site-url=http://testsite:7001
```


sync_product_apns

This utility:

- Enables a site to synchronize product appearance path nodes (APNs) with other sites.
- Synchronizes the absolute occurrence data that is associated with these APNs.
- Synchronizes item revision attachments with the **RDV_appgrp_toplevel_relation** relationship type.

In RDV, this attachment type associates part usage occurrence groups with an architecture revision. Part usage occurrence groups are workspace objects that collect one architecture element (APN) and all corresponding part usage occurrences (APNs). A part usage references its architecture element by means of this part usage occurrence group. Because **RDV_appgrp_toplevel_relation** attachments are handled with this utility, Siemens Digital Industries Software recommends you exclude this relationship when replicating the architecture revision with the **data_sync** or **data_share** utilities.

This utility is similar to the **sync_product_variant_data** utility in that it synchronizes occurrence data, occurrence roots, APNs, and so on, while the **sync_product_variant_data** utility synchronizes variant data, such as variant objects, variant revision objects, item revision expressions, and NVEs.

This utility operates in the following modes and each mode has different mandatory and optional arguments:

Export to disk

Use of the **-item** argument indicates that the utility runs in **export** mode. Note that either **-dir=directory-name** or **-count** are required as indicated with the { | } notation.

Send to remote IDSM after exporting

Use of the **-dir** and **-site** arguments indicate that the utility runs in **send** mode.

Directly send to remote IDSM

Use of the **-send** argument sends the data to the remote IDSM without first exporting to disk.

Read/import from disk

Use of the **-dir** argument without the **-site** argument indicates that the utility runs in **read/import** mode. Note that the utility runs in **read-only** mode if the **-browse** argument is specified, otherwise data is imported to the local site.

SYNTAX

Export to disk

```
sync_product_apns[-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=key-id] [-arch] [-bypass]
-site=site-name [-reason=text]
{-dir=directory-name [-tag_list=file-name [-from=number] [-to=number] ]
| -count[-tag_list=file-name] }
```

```
[-modified_only] [-debug]
[-exclude_variants]
[-silent] [-rollback_on_failure]
```

Send to remote IDSM after exporting

```
sync_product_apns[-u=user-id {-p=password | -pf=password-file} -g=group]
-dir=directory-name -site=site-name
[-from=number] [-to=number]
[-silent] [-rollback_on_failure]
```

Directly send to remote IDSM

```
sync_product_apns[-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=key-id]
-site=site-name[-reason=text]
[-from=number] [-to=number]
{-dir=directory-name [-tag_list=file-name [-from=number] [-to=number]]
| -count[-tag_list=file-name] }
-send
```

Read / import from disk

```
sync_product_apns[-u=user-id {-p=password | -pf=password-file} -g=group]
-dir=directory-name [-browse] [-bypass]
[-from=number] [-to=number]
[-silent] [-rollback_on_failure]
```

ARGUMENTS

-dir

Specifies a local directory name. The utility writes exported data into this directory. If the directory exists, it must not contain export data.

For **export** mode, the directory may or may not exist. If the directory exists, it must not contain exported data. For example, it must not have been used as an export directory in a previous run. For the other modes, the directory must exist and must contain export data.

-item

Item ID of the item of which appearance and absolute occurrence data is to be exported.

-key

Specifies the key of the object. The **-key** argument can be used instead of the **-item** argument. To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-arch

Exports appearance and absolute occurrence data of the architecture items that are associated with the item specified with the **-item** argument.

-site

This argument cannot be used in **read/browse** mode.

Export mode

Specifies the sites to which objects are to be replicated. Upon exporting an object, the utility creates one export record for each site if this object type supports export records. Export records contain the time of export to support incremental replication. If the user requests a roll back at the end of the utility, all export records that were generated during export are rolled back.

Send modes

Specifies the sites to which objects are to be sent. This mode requires an IDSM service to be available at the remote site. If multiple **-site** switches are used to specify multiple sites, the export directory is sent to each site sequentially. For better throughput, run separate processes for each site instead of running one process for multiple sites.

-count

Reports the number of objects to export. If used with the **-modified_only** argument, the count indicates the number of changed objects that need to be replicated to at least one of the sites specified with the **-site** argument. If used with the **-tag_list** argument, a tag list file is written.

-exclude_variants

Causes the export to exclude objects of either the **Variant** or **VariantRevision** class that are attached to the item specified with the **-item** argument. This argument is useful when exporting in multiple batches. If this option is not set, each batch contains all variant options referenced from its expressions causing multiple batches to contain an overlapping set of objects. Overlapping sets of objects cannot be imported in parallel.

-apn_roots_only

Specifies to only export appearance path node roots and absolute occurrence data qualifiers. If this argument is not specified, these objects are excluded.

-silent

Disables rollback capability and rollback confirmation dialog and executes in silent mode.

-rollback_on_failure

Automatically rolls back on any error code other than **ITK_ok**.

-debug

Turns on verbose mode that generates a detailed export and modified dates.

-bypass

Disables access control. You must have DBA privileges to use this argument.

-modified_only

Specifies to only export objects that have changed since they were exported to the sites specified by the **-site** argument.

Note:

If more than one site is specified, the object is exported according to the most out-of-date export record. This can cause some objects to be exported to some sites even though these sites appear to have an up-to-date export record. After completing the export, all sites specified with the **-site** argument have the same export date for each exported object.

-reason

Specifies the reason for export. The maximum number of characters is 240. If no reason is specified, the command line of running the export utility is used as the *reason* text.

-from

Specifies the index of the first object to be exported. The default value is zero.

In **Read/import from disk** mode, the index refers to the object list in the export data located in the directory specified with the **-dir** argument. In other modes, the index refers to the tag list file specified with the **-tag_list** argument.

-to

Specifies the index of the object *following* the last object to process. For example, if you specify **-to=10**, the object at index 9 is the last object to be processed.

In **Read/import from disk** mode, the index refers to the object list in the export data located in the directory specified with the **-dir** argument. In other modes, the index refers to the tag list file specified with the **-tag_list** argument.

-tag_list

Specifies a file name containing a list of object UIDs to export. This file is written if the **-count** argument is specified. If the **-count** argument is not specified, the file is read.

-send

Specifies that the data is sent directly to the remote ISDM.

-browse

Specifies to browse objects in the export area without exporting or importing.

-u

Specifies the user ID. This is a required argument unless the **TC_auto_login** site preference is set. This is a user with administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument. This is a required argument unless the **TC_auto_login** site preference is set.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument. For more information about managing password files, see *Manage password files*.

-g

Specifies the group associated with the user. This is a required argument unless the **TC_auto_login** site preference is set. If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

To bypass the export of named variant expressions, set the **RDV_export_nve** environment variable to **FALSE**.

When using the **data_share** and **data_sync** utilities, Siemens Digital Industries Software recommends you exclude APN and NVE synchronization by setting the following environment variables:

- **TC_EXCLUDE_APN=TRUE**
- **RDV_export_nve=FALSE**

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

1. Export to disk all appearance path nodes, absolute occurrences, and absolute occurrence data on item **RDV00190**, excluding the item itself:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
                  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190
                  -site=cologne -site=turin
```

- Export to disk all appearance path nodes, absolute occurrences, and absolute occurrence data on item **RDV00190**, along with data of associated architecture breakdowns:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190 -arch
  -site=cologne -site=turin
```

- Export to disk all modified appearance path nodes, absolute occurrences, and absolute occurrence data on item **RDV00190**, and its architecture breakdowns:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190 -arch
  -modified_only -site=cologne -site=turin
```

- Determine objects for an incremental update of **RDV00190/D** to sites Cologne and Turin. This includes appearance path nodes, absolute occurrences, and absolute occurrence data for all revisions of **RDV00190** and all revisions of associated architecture breakdown items. Object UIDs are written to the **uid.txt** text file for later export.

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190 -arch
  -count -modified_only -site=cologne -site=turin -tag_list=uid.txt
```

- Export to disk object index numbers 1,000 through 1,999 of all appearance path nodes, absolute occurrences, and absolute occurrence data of item **RDV00190**, along with appearance path nodes, absolute occurrences, and absolute occurrence data of associated architecture breakdowns. Objects are exported as determined in example 4.

The **-exclude_variants** argument causes variant and variant revision objects associated with item **RDV00190** to be excluded. This allows exporting variant data with the **sync_product_variant_data** utility in parallel.

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190 -arch
  -site=cologne -site=turin -from=1000 -to=2000
  -exclude_variants -tag_list=uid.txt
```

- Export to disk object index numbers 1,000 through the end of all appearance path nodes, absolute occurrences, and absolute occurrence data of item **RDV00190**, along with appearance path nodes, absolute occurrences, and absolute occurrence data of associated architecture breakdowns:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
  -bypass -dir=C:\Temp\RDV00190_apns -item=RDV00190 -arch
  -site=cologne -site=turin -from=1000 -exclude_variants
```

- Browse export area:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_apns -browse
```

8. Send export area to Turin's IDSM:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_apns -site=turin
```

9. Send object index numbers 1,000 through 1,999 in export area to Turin:

```
sync_product_apns -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_apns -site=turin -from=1000 -to=2000
```

10. Browse import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_apns -browse
```

11. Import objects in import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_variant_data
```

12. Import object index numbers 1,000 through 1,999 in import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-dir=C:\Temp\RDV00190_apns -from=1000 -to=2000
```

sync_product_variant_data

Enables sites to synchronize product variant data with other sites.

This utility is similar to the **sync_product_apns** utility in that it synchronizes variant data, such as variant objects, variant revision objects, item revision expressions, and NVEs while the **sync_product_apns** utility synchronizes occurrence data, occurrence roots, APNs, and so on.

This utility operates in the following modes and each mode has different mandatory and optional arguments:

Export to disk

Using the **-item** argument indicates that the utility runs in **export** mode. Note that either **-dir=directory-name** or **-count** are required as indicated with the {} notation.

Send to remote IDSM after exporting

Using the **-dir** and **-site** arguments indicate that the utility runs in **send** mode.

Directly send to remote IDSM

Use of the **-send** argument sends the data to the remote IDSM without first exporting to disk.

Read / import from disk

Using the **-dir** argument without the **-site** argument indicates that the utility runs in **read/import** mode. The utility runs in **read-only** mode if the **-browse** argument is specified; otherwise, data is imported to the local site

SYNTAX

Export to disk

```
sync_product_variant_data
[-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=key-id] [-rev=revision-id] [-arch] [-bypass]
-site=site-name [-reason=text]
{-dir=directory-name [-tag_list=file-name] [-from=number] [-to=number] }
| -count[-tag_list=file-name] }
[-modified_only] [-debug]
[-exclude_variants]
[-silent] [-rollback_on_failure]
```

Send to remote IDSM

```
sync_product_variant_data
[-u=user-id -p=password | -pf=password-file -g=group]-dir=directory-name
-site=site-name
[-from=number] [-to=number]
[-silent] [-rollback_on_failure]
```

Directly send to remote IDSM


```

sync_product_variant_data[-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=key-id]
-site=site-name[-reason=text]
[-from=number] [-to=number]
{-dir=directory-name [-tag_list=file-name [-from=number] [-to=number]]
| -count[-tag_list=file-name] }
-send

```

Read / import from disk

```

sync_product_variant_data
[-u=user-id {-p=password | -pf=password-file} -g=group]
-dir=directory-name [-browse] [-bypass]
[-from=number] [-to=number]
[-silent] [-rollback_on_failure]

```

ARGUMENTS

-dir

Specifies a local directory name. The utility writes exported area into this directory. If the directory exists, it must not contain export data.

For **export** mode, the directory may or may not exist. If the directory exists, it must not contain exported data, for example, it must not have been used as an export directory in a previous run. For other modes, the directory must exist and must contain export data.

-item

Specifies the item ID of the item of which variant data is to be exported.

-key

Specifies the key of the object. The **-key** argument can be used instead of the **-item** argument. To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev

Specifies the revision ID of the item revision of which variant data is to be exported in addition to the variant data of its item.

-arch

Exports variant data of the architecture items and architecture revisions that are associated with the item specified with the **-item** argument and the revision specified with the **-rev** argument.

-site

This argument cannot be used in **read/browse** mode.

Export mode

Specifies the sites to which objects are to be replicated. Upon exporting an object, the utility creates one export record for each site if this object type supports export records. Export records

contain the time of export to support incremental replication. If the user requests a rollback at the end of the utility, all export records that were generated during export are rolled back.

Send modes

Specifies the sites to which objects are to be sent. This mode requires an IDSM service to be available at the remote site. If multiple **-site** switches are used to specify multiple sites, the export directory is sent to each site sequentially. For better throughput, run separate processes for each site instead of running one process for multiple sites.

-count

Reports the number of objects to export. If used with the **-modified_only** argument, the count indicates the number of changed objects that need to be replicated to at least one of the sites specified with the **-site** argument. If used with the **-tag_list** argument, a tag list file is written.

-exclude_variants

Causes the export to exclude objects of either the **Variant** or **VariantRevision** class that are attached to the item specified with the **-item** argument. This argument is useful when exporting in multiple batches. If this option is not set, each batch contains all variant options referenced from its expressions causing multiple batches to contain an overlapping set of objects. Overlapping sets of objects cannot be imported in parallel.

-silent

Disables rollback capability and rollback confirmation dialog and executes in silent mode.

-rollback_on_failure

Automatically rolls back on any error code other than **ITK_ok**.

-debug

Turns on verbose mode, which generates a detailed report of export and modified dates.

-bypass

Disables access control. You must have DBA privileges to use this argument.

-modified_only

Specifies to only export objects that have changed since they were exported to the sites specified by the **-site** argument.

Note:

If more than one site is specified, the object is exported according to the most out-of-date export record. This can cause some objects to be exported to some sites even though these sites appear to have an up-to-date export record. After completing the export, all sites specified with the **-site** argument have the same export date for each exported object.

-reason

Specifies the reason for export. The maximum number of characters is 240. If no reason is specified, the command line of running the export utility is used as the *reason* text.

-from

Specifies the index of the first object to be exported. The default value is zero.

In **Read/import from disk** mode, the index refers to the object list in the export data located in the directory specified with the **-dir** argument. In other modes, the index refers to the tag list file specified with the **-tag_list** argument.

-to

Specifies the index of the object *following* the last object to process. For example if you specify **-to=10**, the object at index 9 is the last object to be processed.

In **Read/import from disk** mode, the index refers to the object list in the export data located in the directory specified with the **-dir** argument. In other modes, the index refers to the tag list file specified with the **-tag_list** argument.

-tag_list

Specifies a file name containing a list of object UUIDs to export. This file is written if the **-count** argument is specified. If the **-count** argument is not specified, the file is read.

-count

Reports the number of objects to export. If used with the **-modified_only** argument, the count indicates the number of changed objects that need to be replicated to at least one of the sites specified with the **-site** argument. If used with the **-tag_list** argument, a tag list file is written.

-browse

Specifies to browse objects in the export area without exporting or importing.

-u

Specifies the user ID. This is a user with administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument. For more information about managing password files, see [Manage password files](#).

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

In addition, to export named variant expressions (NVEs), you must set the **RDV_export_nve** preference to **TRUE**. To bypass the export of NVEs, set the **RDV_export_nve** preference to **FALSE**.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

1. Export to disk all variant options and variant revisions of item **RDV00190**, excluding the item itself:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190
-site=cologne -site=turin
```

2. Export to disk all variant options and variant revisions of item **RDV00190** with variant data of associated architecture breakdowns (mostly NVEs):

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190
-arch -site=cologne -site=turin
```

3. Export to disk all variant options and variant revisions of item **RDV00190** and all modified NVEs on its architecture breakdowns:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190
-arch -modified_only -site=cologne -site=turin
```

4. Determine objects for an incremental update of **RDV00190/D** to sites Cologne and Turin. This includes variant expressions of **RDV00190/D** and its associated architecture breakdown revisions (mostly **DECLARE** and **DEFAULT** statements) and the NVEs of associated architecture breakdown items. Object UIDs are written to the **uid.txt** text file for later export.

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190 -rev=D
```

```
-arch -count -modified_only -site=cologne -site=turin
-exclude_variants -tag_list=uid.txt
```

5. Export to disk all variant options of item **RDV00190** and all variant expressions (mostly **DECLARE** and **DEFAULT** statements) on revision D:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190
-rev=D -site=cologne -site=turin
```

6. Export to disk all variant options of item **RDV00190** and all variant expressions (mostly **DECLARE** and **DEFAULT** statements) on revision D, along with variant data of associated architecture breakdowns (mostly NVEs), and associated architecture breakdown revisions (mostly **DECLARE** and **DEFAULT** statements):

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190
-rev=D -arch -site=cologne -site=turin
```

7. Export to disk object index numbers 1,000 through 1,999 of all variant expressions (mostly **DECLARE** and **DEFAULT** statements) of revision D of item **RDV00190**, along with variant data of associated architecture breakdowns (mostly NVEs), associated architecture breakdown revisions (mostly **DECLARE** and **DEFAULT** statements), and their attached saved variant rules. Objects are exported as determined in example 4.

To allow exporting and importing multiple batches in parallel it is recommended to export and import data as described in example 1, before batches as described in this example are exported and imported. The **-exclude_variants** command line option causes the objects exported in example 1 to be excluded. Otherwise, these variants would be exported into each export batch and upon importing a conflict may arise when importing the same variant object via parallel IDSM processes at the same time.

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190 -rev=D
-arch -site=cologne -site=turin -from=1000 -to=2000
-exclude_variants -tag_list=uid.txt
```

8. Export to disk object index numbers 1,000 through the end of all variant expressions (mostly **DECLARE** and **DEFAULT** statements) on revision D of item **RDV00190**, along with variant data of associated architecture breakdowns (mostly NVEs), associated architecture breakdown revisions (mostly **DECLARE** and **DEFAULT** statements), and their attached saved variant rules:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group
-bypass -dir=C:\Temp\RDV00190_variant_data -item=RDV00190 -rev=D
-arch -site=cologne -site=turin -from=1000 -exclude_variants
```

9. Browse export area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data -browse
```

10. Send export area to Turin's IDSM:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data -site=turin
```

11. Send object index numbers 1,000 through 1,999 in export area to Turin:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data -site=turin  
-from=1000 -to=2000
```

12. Browse import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data -browse
```

13. Import objects in import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data
```

14. Import object index numbers 1,000 through 1,999 in import area:

```
sync_product_variant_data -u=Tc-admin-user -p=password -g=group  
-dir=C:\Temp\RDV00190_variant_data -from=1000 -to=2000
```

15. Export all variant options and variant revisions to the remote site:

```
sync_product_variant -u=Tc-admin-user -p=password -g=group  
-send -item=xxx -arch -site=yyyy
```

update_loggeddatetogmt

DESCRIPTION

Run this utility after you migrate Teamcenter from a version prior to 8.3 to version 8.3 or later. This utility updates the values in the **LoggedDate** column from the local time zone to the GMT time zone.

This utility does a bulk update for Oracle and SQL Server databases.

Note:

Run this utility only once. If you run this utility multiple times, the time zone correction is applied multiple times.

This utility may take a long time to run depending on your data size.

SYNTAX

```
update_loggeddatetogmt [-u=user-id {-p=password | -pf=password-file} -g=group]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument cannot be replaced with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

update_objecttype

Run this utility after you migrate Teamcenter from a version prior to 8.3 to version 8.3 or later. This utility updates the **ObjectType** column with the correct Teamcenter object type. The default object type is **POM_object**.

Note:

This utility may take a long time to run depending on your data size.

Running this utility is optional.

SYNTAX

```
update_objecttype [-u=user-id {-p=password | -pf=password-file} -g=group]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument cannot be replaced with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

update_project_bom

Allows you to update all items in a BOM structure within specified projects.

SYNTAX

```
update_project_bom [-u=user-id -p=password | -pf=password-file -g=group]
[-f={add | remove | add_owning_project | change_owning_project}]
[-type= {item | rev}]
[-item=item-id | -key=key-id |
-searchCriteria=Class{attribute1=value1,attribute2=value2,...,attributeN=valueN} |
-file=input_file_path]
[-rev_id=revision-id]
[-rev_rule=revision-rule] [-unit_no=unit-number] [-date=date]
[-end_item=end-item-id | -end_key=end-key-id] [-var_rule=variant-rule]
[-depth=depth-of-bom]
[-level={ 1 | 2 } ] [-projects=project-lists | owning_project=owning-project] [-report=report_file_path]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following types of operation for this utility:

Note:

If this argument is omitted, the default action is to add items to projects.

Note:

The report file is generated only when you use one of the following arguments:

- **-f=add_owning_project**
- **-f=change_owning_project**

add

Adds item objects to projects.

Note:

If this argument is specified, the utility is executed with **Bypass** mode and the access control list rules are bypassed.

remove

Removes item objects from projects.

Note:

If this argument is specified, the utility is executed with **Bypass** mode and the access control list rules are bypassed.

add_owning_project

Specifies adding **owning_project** to objects specified in search criteria/input file.

Note:

If this argument is specified, then **-projects** will be ignored.

Example:

```
update_project_bom -u=Tc-admin-user -p=password -g=group
-f=add_owning_project -searchCriteria=item{item_id=string}
-owning_project=project_id -report=report_file_path
```

change_owning_project

Specifies changing **owning_project** of objects specified in search criteria/input file.

Note:

If this argument is specified, then **-projects** will be ignored.

Example:

```
update_project_bom -u=Tc-admin-user -p=password -g=group
-f=change_owning_project -file=input_file_path
-owning_project=project_id
-report=report_file_path
```

-type

Specifies one of the following types to be updated for each BOM line:

item

Updates each child item for the projects.

rev

Updates only child item revision objects to the projects.

Note:

If this argument is omitted, the default type is **item**.

-item

Specifies the ID of the root item of the BOM to update. Use either the **-item** argument or the **-key** argument.

-key

Specifies the key of the root item of the BOM to update. Use either the **-item** argument or the **-key** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev_id

Specifies the ID of the root item revision. If omitted, the default is the latest revision.

-rev_rule

Specifies the configuration rule to be applied to the item revision. If omitted, the default is the **Latest Working** revision rule.

-unit_no

Specifies the unit number associated with the revision rule.

-date

Specifies the effectivity date associated with the revision rule. The date should be specified in the following format:

yyyy MM dd hh mm ss

-end_item

Specifies the ID of the end item associated with the revision rule. Use with the **-item** argument.

-end_key

Specifies the key of the end item associated with the revision rule. Use with the **-key** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-file

Specifies the plain text file containing the item ids of interested items, listed on separate lines.

Example:

item_000001

item_000002

item_000003

item_000101

Note:

This option can be used with the following arguments:

- **-f=add_owning_project**
- **-f=change_owning_project**

-var_rule

Specifies the variant rule to be applied to the BOM structure. If omitted, no variant rule is applied.

-depth

Specifies to what depth the BOM is traversed. If omitted, the entire BOM structure is traversed.

-level

Indicates the level of propagation. Specify either **1** or **2**.

-projects

Lists the projects to which the BOM structure will be added or from which the BOM structure will be removed.

When the **-add** option is specified, all items in the BOM structure are added to these projects.

When the **-remove** option is specified, all items currently belonging to the projects are removed from the projects. You can specify more than one project. If there is more than one project in the list, each project name is separated by a comma (,).

-report

Specifies the file path to the generated report.

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- To display usage help for this utility, enter the following command on a single line:

```
update_project_bom -h
```

- To generate a report file showing the add owning projects for the objects specified in the search criteria/input file:

```
update_project_bom -u=Tc-admin-user -p=password -g=group
-f=add_owning_project -file=C:\kits\file.txt -owning_project=acme1
-report=C:\kits\report1.log
```

This command generates the following report file:

```
Object Name|Object Type|Status|Comments|Error/Warnings
a|ITEM|OK|a owning_project set to project acme1.
```

- To generate a report file showing the changed owning projects for the objects specified in the search criteria/input file:

```
update_project_bom -u=Tc-admin-user -p=password -g=group
-f=change_owning_project -file=C:\kits\file.txt -owning_project=acme1
-report=C:\kits\report1.log
```

This command generates the following report file:

```
Object Name|Object Type|Status|Comments|Error/Warnings
report1|ITEM|OK|report1 is assigned to project to acme1.
```

- To traverse a BOM structure with the top-level item **ABC001**, item revision **001**, and revision rule **Latest Working**, and to find all child items and add these items to the following three projects: **CusProj1**, **CusProj2**, and **CusProj3**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=add
-item=ABC001 -rev_id=001 -rev_rule="Latest Working"
-projects=CusProj1,CusProj2,CusProj3
```

- To traverse a BOM structure with the top-level item **ABC001**, item revision **001**, and revision rule **Latest Working**, and to find all the child revision items and add only these revision items to projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=add -type=rev
-item=ABC001 -rev_id=001
-rev_rule="Latest Working" -projects=CusProj1,CusProj2
```

- To traverse the BOM structure starting at the top-level item **ABC001** with item revision **001** by applying the revision rule **Latest Released** and variant rule **AlphaRelease**, and find all the child revision items and add only these revision items to the projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=add -type=rev
-item=ABC001 -rev_id=001
-rev_rule="Latest Released" -var_rule="AlphaRelease"
-projects=CusProj1,CusProj2
```

- To traverse the BOM structure of the top-level item **ABC002**, item revision **001**, and revision rule **Latest Working** and find all the child items and remove these items from projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=remove
-item=ABC002 -rev_id=001 -rev_rule="Latest Working"
-projects=CusProj1,CusProj2
```

- To traverse the BOM structure of the top-level item **ABC002**, item revision **001**, and revision rule **Latest working** and find all the child revision items and remove only these revision items from projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=remove
-type=rev -item=ABC002 -rev_id=001
-rev_rule="Latest Working" -projects=CusProj1,CusProj2
```


- To traverse the BOM structure in the top three levels with the top-level item **ABC002**, item revision **001**, and revision rule **Latest working**, and find all the child revision items in the top three levels and remove only these revision items from the projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=Tc-admin-user -p=password -g=group -f=remove
-type=item -item=ABC002 -rev_id=001
-rev_rule="Latest Working" -projects=CusProj1,CusProj2 -depth=3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: locate all the child items in the BOM based on item revision **001**, revision rule **Latest working**, and include these items into the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: no level 2 propagation.

```
update_project_bom -u=Tc-admin-user -p=password -g=group -item=ABC001
-rev_id=001 -rev_rule="Latest Working"
-level=1 -projects=CusProj1,CusProj2,CusProj3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: locate all the child items in the BOM based on item revision **001**, revision rule **Latest working**, and include these items into the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: collect all dataset type objects attached to the BOM line during the BOM traversal. Recursively find all objects that relate to the dataset type objects through the relation specified in the **TC_project_propagate_from_dataset** preference and propagate all of these level 2 objects into the **CusProj1**, **CusProj2**, and **CusProj3** projects.

```
update_project_bom -u=Tc-admin-user -p=password -g=group -item=ABC001
-rev_id=001 -rev_rule="Latest Working" -level=2
-projects=CusProj1,CusProj2,CusProj3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: traverse the BOM based on item revision **001**, revision rule **Latest working**, and variant rule **AlphaRelease**, find all the child revision items and include only these revision items into the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: collect all dataset type objects attached to the revision items in the BOM structure during the level 1 propagation. Recursively locate all objects that relate to the dataset type objects through the relation specified in the **TC_project_propagate_from_dataset** preference and propagate all of these level 2 dependencies into the **CusProj1**, **CusProj2**, and **CusProj3** projects.

```
update_project_bom -u=Tc-admin-user -p=password -g=group -item=ABC001  
-rev_id=001 -rev_rule="Latest Working"  
-var_rule="AlphaRelease" -level=2 -projects=CusProj1,CusProj2,CusProj3
```

upgrade_rev_rules

Upgrades revision rules so that they can safely use the modified revision rule implementation. This is run automatically as part of the upgrade script, but may need to be run again if any locked revision rules are encountered (or if any other error occurs).

SYNTAX

```
upgrade_rev_rules [-u=user-id -p=password | -pf=password-file  
-g=group][-v][-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-v

Verbose mode. Shows additional messages.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

upgrade_variants

Use the **upgrade_variants** utility to upgrade an existing Teamcenter system to the new variant model. It allows you to:

- Migrate the variants for one program (product item) at a time or migrate the entire database.
- Migrate only selected variant conditions and named variant expressions.
- Migrate the design structure or the architecture structure.
- Perform a dry run to validate data integrity before migrating production data.
- Generate a report on all named variant expressions that were migrated.

SYNTAX

```
upgrade_variants [-u=user-id -p=password | -pf=password-file -g=group]
-product_code=product code -ves_list_file=variant expression uid list file
-top_arch_id=top level architecture item -full_database=yes | no
-lou_holder_id=item id of lou holder -file_path=path name
-dry_run=yes | no [-long] -report_delete_failures=yes | no
[-force] -delete_unreferenced_ves_only=yes | no -delete_tree=yes | no
-bulk_delete=yes | no [-parallel] [-h]
```

ARGUMENTS

Note:

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-product_code

Specifies the product code from the product definition form.

-ves_list_file

Specifies a text file containing the UIDs of the variant expressions to migrate.

-top_arch_id

Specifies the item ID of the top level assembly breakdown structure. The named variant expressions for the architecture breakdown are migrated.

-full_database

Specify **yes** to migrate all the variant conditions and NVEs in the entire database or **no** to migrate selected data. The default is **no**.

-lou_holder_id

Specifies the item ID of the LOU holder item. The variant conditions for the LOUs under this LOU holder are migrated.

-file_path

Specifies the path where the migration report should be stored. If no path is specified, the report is stored in the path identified in the **TC_TEMP_DIR** environment variable.

-dry_run

If specified as **yes**, the utility performs a dry run. A dry run does not migrate the variant expressions but generate a migration report. You must specifically set this argument to **no** if you do *not* want to perform a dry run.

-long

Optionally used when you perform a dry run. If specified, the utility outputs UIDs in addition to the variant expression text and named variant expression names.

-report_delete_failures

If specified as **yes**, reports failures encountered while deleting the variant expressions that are not successfully migrated. The default value is **no**.

-force

Optionally forces conversion of all variant expressions to the new variant model.

-delete_unreferenced_ves_only

Optionally deletes unreferenced variant expressions from the database. The default value is **no**.

-delete_tree

Optionally skips deletion of payload trees. The default value is **no**.

-bulk_delete

Optionally performs a bulk deletion of all unreferenced variant expressions. The default value is **no**.

-parallel

Optionally specifies a parallel variant model upgrade session when a session is already in progress.

-h

Displays help for this utility.

ENVIRONMENT

This utility should be run in a shell where the Teamcenter environment is set up.

FILES

A report file is generated in the specified location.

RESTRICTIONS

None.

EXAMPLES

1. To migrate only the assembly structure:

```
upgrade_variants -u=Tc-admin-user -p=password -g=group
-product_code=PLM00001 -file_path=c:\report.txt
-dry_run=no
```

2. To migrate the assembly structure and architecture breakdown (NVEs and LOUs):

```
upgrade_variants -u=Tc-admin-user -p=password -g=group
-product_code=PLM00001 -top_arch_id=PLM0002
-lou_holder_id=PLM00003 -file_path=c:\report.txt -dry_run=no
```

3. To migrate only LOUs in the assembly breakdown:

```
upgrade_variants -u=Tc-admin-user -p=password -g=group  
-lou_holder_id =GM000003 -file_path=c:\report.txt  
-dry_run=no
```

4. To create a variant expressions list file:

```
upgrade_variants -u=Tc-admin-user -p=password -g=group  
-ves_list_file=c:\vesList.txt -file_path=c:\report.txt  
-dry_run=no
```

5. To perform a dry run of a full database migration:

```
upgrade_variants -u=Tc-admin-user -p=password -g=group  
-full_database=yes -file_path=c:\report.txt -dry_run=yes
```

Effectivity mode

effupgrade

Converts effectivity data created in iMAN versions prior to 7.0 into the effectivity model used in iMAN version 7.0 and later, Teamcenter Engineering and Teamcenter. The new effectivity model allows end item qualification and discontinuous ranges. The upgrade process goes through each unconverted release status and creates a 7.0 effectivity qualified against a null end item from the start date, end date or unit values on the release status.

SYNTAX

```
effupgrade [-u=user-id -p=password | -pf=password-file -g=group]  
[-s] | [-i=interval] [-v [-e]] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-s

Specifies a single run. Upgrade runs once and ignores locked statuses.

-i

Specifies multiple runs and an interval between reruns in minutes. The default interval is 60 minutes.

-v

Verbose mode. Shows additional messages.

-e

Displays effectivities of release statuses to be converted. Must be used with the **-v** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- For cases where an upgrade is not going to be disruptive (less than 1000 effectivities), make sure all users are logged off, then upgrade with the **-s** argument. For example:

```
effupgrade -u=Tc-admin-user -p=password -g=group -s -v
```

- For extensive upgrades that are likely to take a long time (many thousands of effectivities), especially if statuses are likely to be locked (some users always logged on), run repetitively until complete.

```
effupgrade -u=Tc-admin-user -p=password -g=group
```



4. 4th Generation Design and Product Configurator utilities

4G Populate utility

Note:

In previous Teamcenter versions, this utility was called **4gd_easy** or **4gd_populate_cd**.

Generates a product design from an existing product or assembly structure. The product design has the following characteristics:

- The name of the product design you specify.
- A set of partitions of a single scheme, that is physical partitions or functional partitions.
- One or more reuse design elements with names derived from the names of the items from which the reuse design elements are realized.

Transformation data is read from each level of the original product structure or assembly. The utility concatenates this data and uses it for design element to collaborative design transformations. Consequently, the collaborative design appears the same as the original assembly. If the original assembly includes absolute occurrences, it uses the absolute occurrence transformations. If the source item revision has bounding boxes attached, the same bounding boxes are attached to the new design element.

By default, the utility configures the assembly with the **Latest Working** revision rule and the latest revision of the source assembly item. Optionally, you can specify a particular revision rule and a item revision. You can also optionally configure the structure by release date or effectivity.

The utility places the new collaborative design in the **Newstuff** folder of the specified user.

SYNTAX

```
4g_populate -u=user-id {-p=password | -pf=password-file} [-g=group]
-i=item_id [-cd_type=type] [-preview_mode] [-preview_file_name=preview file name]
[-create_mode] [-rev=revision] [-rule=revision_rule]
[-date_effectivity=date_effectivity] [-end_item_id=end_item_id]
[-release_date=release_date] [-map_file=file_name] [-batch_size=size]
[-split_mode] [-number_of_batches=number_of_batches] [-split_directory=directory_path]
[-pre_validate_mode] [-pre_validation_report_file_name=file_path_name]
[-child_assembly_validation] [-update] [-updateOfDesignChanges] [-h]
```

ARGUMENTS

General options:

-u= Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p=	Specifies the password.
	This argument is mutually exclusive with the -pf argument.
-pf=	Specifies the password file.
	For more information about managing password files, see Manage password files .
	This argument is mutually exclusive with the -p argument.
-g=	Specifies the group associated with the user.
	If used without a value, the user's default group is assumed.
-i=	Specifies the item ID of the precise product structure or assembly to realize.
-rev=revision (optional)	Specifies the revision of the specified item. Default: Latest .
-rule=revision rule (optional)	Specifies the name of the revision rule to configure the source assembly. If the name of the rule contains spaces, you must enclose it in quotation marks, that is, " rev rule name ". Default: Latest Working .
-cd_type=type (optional)	Specifies the type of the product design. If a custom type is not specified, the type of product design created is collaborative design .
-date-effectivity=date (optional)	Specifies a date in string format (<i>dd-mmm-yy</i>) which must be appended to the specified revision rule.
-end_item_id = end_item_id (optional)	Specifies the item ID of the end item which must be appended to the specified revision rule.
-release_date = release_date (optional)	Specifies the release date in string format (<i>DD-MMM-YY</i>) which must be appended to the specified revision rule.

- preview_mode** Enables preview mode.
- create_mode** Enables create mode.
- preview_file_name=file_name** Specifies the full path of the preview file to create or update. Only valid if you enable preview mode.
- map_file=file_name** Specifies the full path of the existing mapping file that you created using the 4G Populate interface.
- batch_size=size** Defines the batch size to be used while creating 4GD objects.
- split_mode** Enables split mode. In split mode, multiple preview files are created.
- number_of_batches=number_of_batches** Determines the number of preview files created in split mode. The number of files is equal to the number of the design elements in the preview file divided by the number of batches.
- split_directory=directory_path** Full path to an existing directory, empty folder where split preview files will be created. You must have write access to this directory.
- pre_validation_mode** Enables pre-validation. Default: **false**
- pre_validation_report_file_name=file_path_name** Prevalidation report file system path including the file name.
- child_assembly_validation (optional)** Designates the entire child assembly for validation.
- update** Enables update mode.
- updateOfDesignChanges (optional)** Updates existing realizations of reuse design elements in the collaborative design. This option may be used to synchronize subordinate level changes if you do not want to deploy automatic synchronization.
- h (optional)** Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in *Log files produced by Teamcenter*.

Results are written to the **bom_output.txt** file in the **c:\temp** directory on Windows systems or **/tmp** directory on Linux systems. It lists the shape and reuse elements that are created, together with the total number of objects created. It also contains the time of the last run and total memory consumed.

RESTRICTIONS

None.

EXAMPLES

```
4g_populate -u=user1 -p=user1 -g=designer -pre_validate_mode -i=031080
-map_file=C:\map_files\generic_car_mapping.xml

-pre_validation_report_file_name=C:\validation_files\pre_validation_report.xml
```

The example above creates a validation report.

```
4g_populate -u=user1 -p=user1 -g=designer
-i=flip_fone_assembly -cd_name=flip_fone_CD
-preview_mode
-preview_file_name=C:\preview_files\flipfone_assembly_preview.xml
-map_file=C:\map_files\flipfone_assembly_mapping.xml
```

The example above creates or updates the **flipfone_assembly_preview.xml** preview file. You must create an **Easy Preview** dataset in Teamcenter and import the generated preview file to preview the product design using the 4G Populate interface.

```
4g_populate -u=user1 -p=user1 -g=designer
-i=031124 -cd_name=CD02
-preview_mode
-cd_type=C2Cpd0CollaborativeDesign
-preview_file_name=C:\preview_files\assy_preview.xml
-map_file=C:\map_files\assy_mapping.xml
```

The example above previews the **C2Cpd0CollaborativeDesign** product design.

```
4g_populate -u=user1 -p=user1 -g=designer
-i=031124 -cd_name=CD02
-preview_mode
-date_effectivity=01-Sep-2017
-preview_file_name=C:\preview_files\assy_preview.xml
-map_file=C:\map_files\assy_mapping.xml
```

The example above previews the **CD02** product design effective 01-Sep-2017.

```
4g_populate -u=user1 -p=user1 -g=designer
-i=031124 -cd_name=CD02
-preview_mode
-released_date=01-Sep-2017
-preview_file_name=C:\preview_files\assy_preview.xml
-map_file=C:\map_files\assy_mapping.xml
```

The example above previews the **CD02** product design that was released on 01-Sep-2017.

```
4g_populate -u=user1 -p=user1 -g=designer
-preview_file_name=C:\preview_files\flipfone_assembly_preview.xml
-split_mode
-split_directory=C:\preview_files\split_flipfone_preview_directory
-number_of_batches=3
```

The example above splits the **flipfone_assembly_preview.xml** preview files into smaller preview files that can be executed simultaneously by the 4G Populate utility when creating the product design

```
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\flipfone_assembly_preview.xml
```

The example above creates the **flip_fone_CD** 4GD product design in Teamcenter from the existing preview file.

```
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\0_Org_flipfone100.xml
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\1_batch_flipfone100.xml
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\2_batch_flipfone100.xml
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\3_batch_flipfone100.xml
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\4_batch_flipfone100.xml
4g_populate -u=user1 -p=user1 -g=designer
-create_mode
-preview_file_name=C:\preview_files\5_batch_flipfone100.xml
```

The example above creates the **flip_fone_CD** 4GD product design in Teamcenter from existing split preview file.

ERROR REPORTS

By default, if you execute the **4g_populate** with the **-create_mode** and **-updateofDesignChanges** arguments, a consolidated error report in the XML format at the location specified in the **TC_TMP_DIR** environment variable. If you do not set this environment variable, the error report is generated at the location from where the **4g_populate** utility was executed.

If you set the **-split_mode** argument, multiple error reports are generated.

You can set the file name of the report in the **4GPopulate_ErrorReport_FileName** environment variable. On setting this environment variable, a consolidated error report is generated for all the **4g_populate** executions for a session. You can set this environment variable either as a user or as a system variable. If you set the environment variable from a command prompt, the consolidated error report is generated till you close the command prompt. If you do not specify the complete path in this environment variable, the error report is generated in the folder from where the utility was executed. While specifying the complete path, do not include any spaces and quotation marks.

In the consolidated error report, the latest run is always added as the first element. To view the error report, open it in any browser.

Sample consolidated error report —

```
<?xml version="1.0"?>
- <Populate4GErrorReport>
  - <Report_05_03_14_14_11_2019
    Preview="C:\Users\xbab4z\AppData\Local\Temp\0_Org_flipfone_assembly.xml" Failed="1">
      - <Failed>
        <Element1 sTargetParentPartitionId="" Target4GObjectID="PTN000010" SourceItemRevId="A"
          SourceItemId="flipfone_subassy_top" Error="A Business Object "Functional Partition" already
            exists with properties "{ID = PTN000010}, {Partition Scheme Type = Function}, {Model
              Object = CD_19jun1617;1-CD_19jun1617}, {Revision ID = 001}". Please specify at least
                one unique value to create the Business Object "Functional Partition"."/>
        </Failed>
      </Report_05_03_14_14_11_2019>
    <Report_56_02_14_14_11_2019
      Preview="C:\Users\xbab4z\AppData\Local\Temp\1_batch_flipfone_assembly.xml" Failed="0"/>
    <Report_51_02_14_14_11_2019
      Preview="C:\Users\xbab4z\AppData\Local\Temp\0_Org_flipfone_assembly.xml" Failed="0"/>
  - <Report_49_01_14_14_11_2019
    Preview="C:\Users\xbab4z\AppData\Local\Temp\0_Org_flipfone_assembly.xml" Failed="1">
      - <Failed>
        <Element1 sTargetParentPartitionId="" Target4GObjectID="PTN000005" SourceItemRevId="A"
          SourceItemId="flipfone_subassy_top" Error="A Business Object "Functional Partition" already
            exists with properties "{ID = PTN000005}, {Partition Scheme Type = Function}, {Model
              Object = CD_19jun1616;1-CD_19jun1616}, {Revision ID = 001}". Please specify at least
                one unique value to create the Business Object "Functional Partition"."/>
        </Failed>
      </Report_49_01_14_14_11_2019>
    </Populate4GErrorReport>
```

Report attribute	Description
Preview	Contains the path of the file used in the 4g_populate utility.
Failed	Specifies the number of elements with errors.
Failed Element	Contains the details related to the failed execution.

availability_to_availability_rule_utility

Creates equivalent availability rules for present availability objects in the system.

Beginning in Teamcenter 10.1.7, Product Configurator on Rich Client — Usage does not consider constraints based on **Family Availability** and **Option Value Availability** objects. To ensure consistent constraint evaluation results, you must generate **Availability Rule** objects that correspond to the **Family Availability** and **Option Value Availability** objects created in earlier Teamcenter versions.

Beginning in Teamcenter 10.1.7, **Family Availability** and **Option Value Availability** objects can no longer be created in Product Configurator on Rich Client — Usage. After the Teamcenter 10.1.7 upgrade, site administrators execute the **availability_to_availability_rule_utility** utility to generate the corresponding **Family Availability** rule objects.

You must run this utility if:

- You are migrating to Teamcenter 10.1.7, and
- **Family Availability** and **Option Value Availability** objects exist in the Teamcenter version you are migrating from.

The **availability_to_availability_rule_utility** utility does not release the newly created **Availability Rule** objects, even if existing **Option Value Availability** or **Family Availability** objects are released. If required, you need to release newly created availability objects manually, after running this utility.

The history of availability objects is not migrated.

The **availability_to_availability_rule_utility** utility execution may take a significant time, depending on the number of existing availability constraints defined in the system. If you have a large number of availability constraints, consider running the utility during off-peak hours.

SYNTAX

```
availability_to_availability_rule_utility [-u=user-id {-p=password -g=group-batch_size}]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

- p**
Specifies the password.
- g**
Specifies the group associated with the user.
- If used without a value, the user's default group is assumed.

-batch_size
Specifies how many rules are processed in bulk. By default, the size is set to 100.

This utility retains the original effectivity as set on the availability constraints prior to migration. Effectivity on availability constraints are applied to newly created availability rules.

Example:

You have a **Color** family with **Red**, **Green**, and **Black** values.

Below are existing availability constraints in Teamcenter 10.1.6.

Family availability constraint	Color → VXI	Color family is made available to model VXI
Value availability constraint	Red → VXI	Red value made available to model VXI
Value availability constraint	Green → VXI	Green value made available to model VXI with Date Effectivity (2012-09-01 to 2016-09-01)

After you run the **availability_to_availability_rule_utility** migration utility, after the Teamcenter 10.1.7 upgrade, two new availability rules are created as follows:

Availability rule	Red → VXI	Applicability → VXI and Subject → Red
Availability rule	Green → VXI	Applicability → VXI and Subject → Green with Date Effectivity (2012-09-01 to 2016-09-01)

manage_effectivity_options

Allows you to manage 4GD effectivity options from a command line.

SYNTAX

```
manage_effectivity_options [-u=user-id {-p=password | -pf=password-file} -g=group]
{-contextItemId=ItemID | -contextItemKey=Key -contextRevisionId=revisionID}
-effOpt=option-name-string
[-effValues=valid-values]
-owningItemId=ItemID | -owningItemKey=Key} | -file=xml input file
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-contextItemId

Specifies the item ID to which to attach a new option. This is typically the item ID of the product for which the effectivity option should be added or modified. The effectivity option may be an effectivity stream group such as Engineering Intent or Model Year.

-contextItemKey

Specifies keys and values to pass if the item ID and revision ID are insufficient to uniquely identify the context item revision. You can specify **contextItemId** or **contextItemKey**, not both.

-contextRevisionId

Specifies the item revision for which a new effectivity option should be attached or modified. The item revision represents the product namespace in which the option will be available.

-effOpt

Identifies the effectivity option to create or modify. This option usually represents an effectivity stream grouping such as Engineering Intent or Model Year.

-effValues

Specifies a comma-separated list of valid values for the effectivity option. This is an absolute list and *all* valid values must be passed. Teamcenter treats the absence of valid values when you update existing effectivity options as a request to remove those values from the context revision. Valid values must be given in the context of the context item revision. You can attach the modified item revision to multiple models for which effectivity streams with these valid values are needed.

-owningItemId

Specifies the item that defines the effectivity option to modify. If specified, the effectivity option defined on the owning item is reused on the product item. It is restricted to the set of specified values if **-effValues** is populated with a list of valid values.

-owningItemKey

Specifies keys and values to pass if the owning item ID is insufficient to uniquely identify the owning item. You can specify **owningItemId** or **owningItemKey**, not both.

-file

Facilitates bulk support of this utility and specifies the name of the XML file that contains all required information. In presence of this argument, all other required arguments, such as **contextItemId** | **contextItemKey**, **contextRevisionId**, **owningItemId** | **owningItemKey**, **effOpt**, **effValues**, are ignored because they are considered mutually exclusive.

Note:

The full file path is required with the file name. If not specified, the current directory is used.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- This example declares effectivity options in a dictionary.

```
manage_effectivity_options -u=Tc_admin -p=password
-contextItemId= DictID -contextRevisionId=A
-effOpt=OptName -effValues=Val1,Val2,Val3
-owningItemId=DictID
```

- This example allocates subset of effectivity options for a configurator context.

```
manage_effectivity_options -u=Tc_admin -p=password
-contextItemId=ContextID -contextRevisionId=A
-effOpt=OptName -effValues= Val1,Val2 -owningItemId= DictID
```

- This example performs bulk update (declaration/allocation).

```
manage_effectivity_options -u=Tc_admin -p=password
-file=<directory path>\ManageEffectivityOptions.xml
```

save_variant_rule_as_variant_criteria

Saves existing variant rules as new variant criteria objects.

All **IMAN Relations** with **VariantRules** are copied. The new **VariantCriteria** objects are saved in the **Newstuff** folder. Existing **VariantRules** remain unchanged.

If successful, the new relations for the new **VariantCriteria** objects are created in the system. Additionally, **IMAN Relations** of the existing objects are deleted based on the user input. If the creation and deletion fails, the system generates a failure report in the log file.

SYNTAX

```
save_variant_rule_as_variant_criteria -u=user-id -p=password -g=group
-file=Input_File_with_Comma_Separated_VariantRule_UIDs -deleteRelations=true_or_false
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Mandatory input. Path to a file with a list of comma separated UUIDs of **VariantRules** to be saved as **VariantCriteria**.

-deleteRelations

(Optional) If set to true, the system deletes **IMAN Relations** of existing **VariantRule** objects if **IMAN Relations** for the new objects are successfully saved in the database. In case of the failure, it is reported in the log file. The default value is false.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

- *Input to the utility*

A CSV file with a comma separated list of UIDs of **VariantRules** to be saved as **VariantCriteria**.

- *Output of the utility*

The **VariantCriteria** objects are created corresponding to the input **VariantRules**. The system generates a log file with the UID and the name of **VariantRule** and the corresponding UID, name, and ID of the newly created **VariantCriteria**. Additionally, if the user specified `deleteRelations = true`, the system generates a log of UIDs of **VariantCriteria**, if any, for which the creation and deletion of **IMAN Relations** have failed.

RESTRICTIONS

None.

EXAMPLES

- *Example 1*

Variant rule UIDs listed in **ListOfUids.txt** are saved as variant criteria.

```
save_variant_rule_as_variant_criteria -u=Tc_admin -p=password -file=ListOfUids.txt
```

- *Example 2*

Variant rule UIDs listed in **ListOfUids.txt** are saved as variant criteria. **IMAN Relations** of old variant rule objects are deleted and **IMAN Relations** to the new variant criteria are saved.

```
save_variant_rule_as_variant_criteria -u=Tc_admin -p=password -file=ListOfUids.txt
-deleteRelations=true
```

ptn0_persist_dynamicMembers

Traverses the complete partition breakdown of a collaborative design, executes every partition recipe, and creates static membership objects in the database. Traversal of a large product may take a significant time and Siemens Digital Industries Software recommends you schedule the utility to execute periodically as a cron job. Execute it at a frequency that ensures the saved state of dynamic members is sufficiently up-to-date to produce reliable results from the **wherePartitioned** service (this service uses static membership objects to identify owning partitions for design elements).

When you run this utility, it deletes membership objects it previously created and creates new membership objects, according to the latest recipes on the partitions. It also generates an error report in the XML format at the location specified in the `TC_TMP_DIR` environment variable. If you do not set this environment variable, the error report is generated at the location from where the **ptn0_persist_dynamicMembers** utility was executed.

Instead of generating a new error report every time this utility is executed, if you want a consolidated error report, set the file name of the report in the **Ptn0PersistDynamicMembers_ErrorReport_FileName** environment variable. You can set this environment variable either as a user or as a system variable. While specifying the file name, do not include any spaces and quotation marks. If you set the environment variable from a command prompt, the consolidated error report is generated till you close the command prompt.

SYNTAX

```
ptn0_persist_dynamicMembers [-u=user-id {-p=password | -pf=password-file}
-g=group] -m=model_id -s=partition scheme type name [-revisionrule=rev rule name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-m

Specifies the identifier of the collaborative design (model) to traverse.

-s

Specifies the type name of the partition scheme, for example, **Ptn0SchemeFunctional**, **Ptn0SchemePhysical**, or **Ptn0SchemeSpatial**.

-revisionrule

Optionally specifies a revision rule to apply. If no revision rule is specified, the utility uses the default revision rule.

-h

Displays help for this utility.

RESTRICTIONS

This utility must be run by a user with Teamcenter administration privileges.

EXAMPLES

```
ptn0_persist_dynamicMembers -u=adminjones -p=passjones -g=dba
-m=new_ship_design -s=Ptn0SchemeFunctional -revisionrule=Any
Status;Working
```

In case of an error, to troubleshoot it, open the error report in any browser.

Sample report with updated failed partitions —



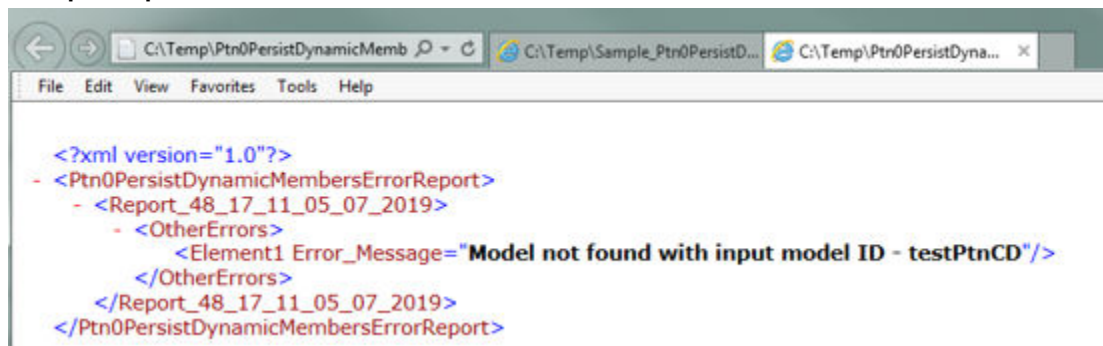
```

<?xml version="1.0"?>
- <Ptn0PersistDynamicMembersErrorReport>
  - <Report_52_52_10_05_07_2019>
    - <ProductDesign1 Update_Failed_Count="2" Display_Name="testPtnCD;1-testPtnCD">
      <Partition1 Display_Name="zoneptn2/001;1-zoneptn2" Error_Message="The "add" operation could not be performed, because the target Partition zoneptn2/001;1-zoneptn2 is inactive: Please check the Teamcenter server syslog file for more information."/>
      <Partition2 Display_Name="zoneptn3/001;1-zoneptn3" Error_Message="The "add" operation could not be performed, because the target Partition zoneptn3/001;1-zoneptn3 is inactive: Please check the Teamcenter server syslog file for more information."/>
    </ProductDesign1>
  </Report_52_52_10_05_07_2019>
</Ptn0PersistDynamicMembersErrorReport>

```

Report attribute	Description
Display_Name (of ProductDesign1)	Contains the display name of the product design element.
Update_Failed_Count (of ProductDesign1)	Specifies the number of partitions that failed to update.
Display_Name (of Partitionpartition_number)	Contains the display name of the partition.
Error_Message (of Partitionpartition_number)	Specifies why the partition failed to update.

Sample report with OtherErrors Element —



```

<?xml version="1.0"?>
- <Ptn0PersistDynamicMembersErrorReport>
  - <Report_48_17_11_05_07_2019>
    - <OtherErrors>
      <Element1 Error_Message="Model not found with input model ID - testPtnCD"/>
    </OtherErrors>
  </Report_48_17_11_05_07_2019>
</Ptn0PersistDynamicMembersErrorReport>

```

Ptn0_set_is_partition_owned_true

Sets the **ptn0is_partition_owned** property on partition memberships, if you are upgrading to Teamcenter 10.1 or later from an earlier version. This property must be set if you want to export or import partition members using TC XML.

SYNTAX

```
Ptn0_set_is_partition_owned_true [-u=user-id {-p=password | -pf=password-file} -g=group] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility must be run by a user with Teamcenter administration privileges.

upgrade_variant_rule_data

Migrates the variant rule objects to the new expression data model that allows to persist the session information.

The perspective parameters and solver profile parameters together constitute the configurator session information for Product Configurator.

This utility is by default run during Teamcenter upgrade process. However, if needed this utility can be run independently.

SYNTAX

```
upgrade_variant_rule_data [-u=user-id {-p=password | -pf=password-file} -g=group]
[-batchSize=batch_size]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-batchSize

Specifies the number of variant rule objects migrated to the new expression data model in each batch. The default batch size is **1000**.

-h

Displays help for this utility.

Note:

While performing the upgrade in the replica site, the variant rule object is not upgraded to the new expression data model. This is done to ensure that site ownership is not changed for the variant rule object.

You should run the **data_sync** utility before using the variant rule object in the replica site. It upgrades the variant rule to the new expression data model.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

• Example 1

In this example, all variant rules that are referring to the old expression data model is migrated to the new expression data model in multiple batches with the default batch size of **1000**.

```
upgrade_variant_rule_data -u=Tc_admin -p=password
```

• Example 2

In this example, all variant rules that are referring to the old expression data model is migrated to the new expression data model in multiple batches with the default batch size of **2000**.

```
upgrade_variant_rule_data -u=Tc_admin -p=password -batchSize=2000
```


validate_revrule_effectivity

Validates the effectivity criteria on one or more specified revision rules, generates validation records, and associates them with the revision rules. Optionally, you can create a text file containing the names of revision rules to validate.

Validation may take a significant time, depending on the number of constraints and default rules defined in the system. If you have a large number of constraints and rules, consider running the utility during off-peak hours.

SYNTAX

```
validate_revrule_effectivity [-u=user-id {-p=password | -pf=password-file} -g=group]
{-rule_name=rule-name1 [-rule_name=rule-name2] | -rev_rule_names_file=file-name}
[-apply_constraints= 0 | 1+] [-apply default= 0 | 1+]
{-model_id=model-id | [-product_name=product-name
-product_namespace=product-namespace]} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-rule_name

Specifies the name of the revision rule to validate. If you want to validate more than one revision rule, use this argument once for each revision rule. This argument is optional if you use the **-rev_rules_file_name** argument, but takes precedence if both arguments are entered.

-rev_rules_file_name

Specifies the name and full path of a text file containing the names of revision rules to validate. The file must contain the name of one revision rule on each line. This argument is optional if you use the **-rule_name** argument.

-apply_constraints

Specifies whether the system applies constraints.

0

The system does not apply constraints.

1

The system applies constraints.

-apply_defaults

Specifies whether the system applies defaults.

0

The system does not apply defaults.

1

The system applies defaults.

-model_id

This is an optional argument and is specified only if you connect to an external configurator. Specifies the identifier of the 4GD model object in the context of which the revision rules should be validated. CPD model objects define the appropriate external configurator using the **EffectivityInModel** object.

-product_name

Specifies the name of the product, for example, the item ID. Teamcenter uses this value in conjunction with the **-product_namespace** argument to resolve any ambiguities in effectivity option value names. If you do not specify a value, Teamcenter deduces the product name from the **EffectivityInModel** object associated with the model.

-product_namespace

Specifies the namespace of the product in which the product name has unique semantics, for example, the item revision ID, model year, and product type. Teamcenter uses this value in conjunction with the **-product_name** argument to resolve any ambiguities in effectivity option

value names. If you do not specify a value, Teamcenter deduces the product namespace from the **EffectivityInModel** object associated with the model.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Consider the following two products that are configured with effectivity:

Product	Default	Rule checks
P1	WHILE Unit > 5 DEFAULT Date >= 2012-01-01	WHILE Unit <= 5 RAISE ERROR "message" IF Date < 2012-01-01
P2		RAISE ERROR "message" Unit > 5

If you validate a specified revision rule with an effectivity criteria of **Unit=7**, you obtain the following results:

Revision rule effectivity criteria	Product	Result	Violations	Applied default
Unit=7	P1	Unit=7, Date=2012-01-01..	None	Date >= 2012-01-01
	P2	Unit=7	Unit > 5: "message"	None

These validation results are persisted in records that are attached to the revision rule. If a client queries the revision rule effectivity criteria in the context of product P1 or P2, the server response includes the corresponding validation records. This allows sites to run this utility to perform potentially CPU intensive criteria validation once for each revision rule during off-peak hours, while permitting the system to deliver an acceptable validation service to hundreds of users during working hours.

There are two limitations to this practice:

- In this version of Teamcenter, you cannot create, view, or change defaults and rule check constraints in the user interface. You can create them only with ITK APIs.

- In this version of Teamcenter, you cannot review validation results in the user interface when asking the server for revision rule effectivity criteria. You can view results only with the **validate_revrule_effectivity** utility.

Also, when you validate the effectivity criteria of a revision rule, the criteria are sent to a configurator service for validation. These criteria may be sent to external or remote configurator services in a future version of Teamcenter (this capability is not supported in the current version). In this case, the construction of the validation records is subject to any limitations of the external or remote configurator service.

EXAMPLES

- This example validates a revision rule called **My Latest Released** in the context of a model whose ID is **ship123**. It does not apply constraints or default rules. It creates a validation record and associates it with the revision rule.

```
validate_revrule_effectivity -u=Tc-admin-user -p=password -g=group
-rule_name=My Latest Released -model_id=ship123
```

- This example validates revision rules called **My Latest Working** and **My Latest Released** which are listed in the file **C:\xyz\revrule.txt** in the context of a model whose ID is **Malibu_V6_2010**. It applies defaults and constraints. It creates validation records for each revision rule and associates them with the relevant revision rules.

```
validate_revrule_effectivity -u=Tc-admin-user -p=password -g=group
-rev_rules_file_name=C:\xyz\revrule.txt
-apply_constraints=1 -apply_defaults=1 -model_id=Malibu_V6_2010
```



5. Workflow utilities

cleanup_userinbox

Consolidates two or more **My Worklist** inboxes of a user into a single inbox, preserving the resource pool subscriptions and surrogate lists of each. If two inboxes have the same surrogate, it is indeterminate which effective start and end dates are kept.

Note:

Under normal circumstances, a Teamcenter user should not have more than a single unique **My Worklist** inbox.

SYNTAX

```
cleanup_userinbox [-u=user-id {-p=password | -pf=password-file} -g=group-name]
[-report=file-name] [-dryrun] [-debug] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-report

Specifies the output file for the report. If this argument is not specified, it is written to the standard output.

-dryrun

Generates a report of the changes to be made but does not update the database. Actions that would be performed are displayed.

-debug

Displays extra diagnostic information. This argument also forces the use of the **-v** argument.

-v

Verbose mode. Provides information about the **My Worklist** inboxes processed. This argument is recommended.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To generate a report about which changes would be made without actually updating the database:

```
cleanup_userinbox -v -dryrun
```

- To consolidate the **tcadmin** user's inboxes and display the report in the standard output:

```
cleanup_userinbox -u=tcadmin -p=password
```

- To consolidate the **tcadmin** user's inboxes and display the report in the **rpt_file** file:

```
cleanup_userinbox -u=tcadmin -pf=pswd_file -report=rpt_file
```


clear_process_stage_list

Clears the **process_stage_list** field of the workspace object.

Note:

Running this utility changes the date and time stamp of the objects that it is run against.

SYNTAX

```
clear_process_stage_list [-u=user-id {-p=password | -pf=password-file} -g=dba] -folder=folder-name [-h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. The group value must be **dba** to run this utility.

-folder

Specifies the folder name where the target workspace objects should be placed whose process stage lists are to be cleared. The folder must be a single folder directly inside the executing user's **Home** folder.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

You must be logged on as a member of the **dba** group.

EXAMPLES

To clear the **process_stage_list** fields of all the objects in the **my_folder** object, enter the following command on a single line:

```
$TC_ROOT/bin/clear_process_stage_list -u=user-id -p=password -g=dba  
-folder=my_folder
```

global_transfer

Transfers all tasks of one user ID or resource pool to another user ID or resource pool. This utility provides the capability to transfer tasks, as follows:

- Users can transfer their own tasks to another user.
- Users can transfer the tasks of other users.
- Users with group administrator privileges can transfer tasks assigned to members of their group.
- System administrators can transfer tasks belonging to any user to a different user.

When transferring tasks, such as do tasks or **select-signoff-team** tasks, the responsible party for each task is transferred to the new resource pool or user.

When transferring **perform-signoff** tasks, the tasks of the current resource pool or current user are delegated to a new resource pool or user if the new resource pool or user meets the same requirements as if the task were delegated using the delegate feature in the Teamcenter interface. If the signoff task is associated with a signoff profile, the delegation is constrained to another user or resource pool of the group/role specified by the signoff profile and the list of users to select from is filtered. If the signoff task is not associated with a signoff profile, delegation to any user or resource pool is possible, and the list of users to select from is not filtered.

SYNTAX

```
global_transfer [-u=user-id {-p=password | -pf=password-file} -g=group-name] [-f=user-ID] [-t=user-ID] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the ID of the user whose inbox tasks are being transferred or the group/role resource pool inbox tasks of a specified group/role. **Group/Role** transfers resource pool inbox tasks for a specified group/role. **Group/*** transfers resource pool inbox tasks of a specified group and any role. ***/Role** transfers resource pool inbox tasks of a specified role and any group.

-t

Specifies the ID of the user to whom the tasks are being transferred, or the group/role transfers resource pool inbox tasks of a specified group/role. **Group/*** transfers resource pool inbox tasks of a specified group and any role. ***/Role** transfers resource pool inbox tasks of a specified role and any group.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

The **gtransfer_XXXX.log** file provides a listing of selected tasks, whether they have been transferred (Y/N), to whom they were transferred, and if there were any errors in the transfer.

RESTRICTIONS

None.

EXAMPLES

To transfer all tasks from user Mike to user Kevin, enter the following command on a single line:

```
global_transfer -f=mike -t=kevin
```

install_handlers

Defines action handlers. It can also configure new action handlers and modify the definition of existing handlers.

SYNTAX

```
install_handlers [-u=user-id {-p=password | -pf=password-file} -g=group-name]
-f= {install | create | modify | delete | listall} -id=handler-ID
[-funcname=function-name]
[-libname=library name]
[-functype=1 | 2]
[-execmode=1 | 2 -desc=handler-description]
[-retrycount=retry-count] [-retryinterval=retry-interval-in-minutes]
[-exectime=time-of-the-day-in-24-hour-format] -override=true | false [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the mode in which the utility must execute. The mode must be one of the following:

- **install**
- **create**
- **modify**
- **delete**
- **listall**

-id=

Specifies the handler ID.

-funcname

Specifies the name of the library function. Use this argument when the **-functype** argument is set to 1.

-libname

Specify the name of the library containing the **-funcname** API. This is optional. If not specified, the following order of libraries is used:

libuser_exits

libsub_mgr

libepm

libsa

-functype

Specifies whether the handler is a library function or a stand-alone executable. The value for this argument must be one of the following:

1

Library function

2

Stand-alone executable

-execmode

Specifies the handler's execution mode. The value for this argument must be one of the following:

1

Executes the handler in the calling process.

2

Executes the handler as a separate process.

-desc

Specifies the handler description.

-override

Specifies if the handler execution time can be overridden. The value for this argument must be one of the following:

true

Allows override.

false

Disables override.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To install the default action handlers, enter the following command on a single line:

```
install_handlers -f=install
```

- To create an action handler with the specified attribute values to set the execution time for the handler to 6.00 p.m., enter the following command on a single line:

Note:

If no library name is specified, the default libraries listed previously are searched.

```
install_handlers -f=create -id=MyActionHandler -funcname=Myfunc
-f=funcname=1 -execmode=1 exectime=1800
```

- To create an action handler with specified attribute values, an execution time set for 6.00 PM, and run from the specified **libmdo** library, enter the following command on a single line:

```
install_handlers -f=create -id=MyActionHandler -funcname=Myfunc
-f=funcname=1 -execmode=1 exectime=1800 -libname=libmdo
```

- To set the retry count value to **5** for the **MyActionHandler** action handler, enter the following command on a single line:

```
install_handlers -f=modify -id=MyActionHandler -retryCount=5
```

- To delete the **MyActionHandler** action handler, enter the following command on a single line:

```
install_handlers -f=delete -ID=MyActionHandler
```

- To list all the action handlers defined in the database, enter the following command on a single line:

```
install_handlers -f=listall
```

migrate_wf_attachments

Transforms workflow process attachments from VLA to GRM relationships.

Active processes are normally migrated during the upgrade from versions prior to Teamcenter 11.2. If you want completed processes to be migrated, run this utility manually after upgrade.

This utility slows down as it loads processes in memory, so you can control the number of migrations by limiting the size of the batch per process with the **-batch_size** argument. When the original process has reached the specified batch size, it starts another process and then terminates itself. The subprocess continues where its parent process left off and then when it reaches its own batch size, it creates another subprocess and so on. You can limit the number of processes running concurrently to improve performance with the **-num_procs** argument .

SYNTAX

```
migrate_wf_attachments [-u=user-id {-p=password | -pf=password-file} -g=group-name] [-process=active | completed | all] [-report] [-batch_size=number-to-upgrade] [-num_procs=number-to-run-concurrently] [-h]
```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

If used without a value or if neither the **-pf** nor the **-p** argument is used, the system displays an error and asks for a username and password interactively.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

If used without a value or if neither the **-pf** nor the **-p** argument is used, the system displays an error and asks for a username and password interactively.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-process

(Optional) Specifies if the attachments in active, completed, or all workflow processes are to be migrated. The allowed values are:

- **active**
- **completed**
- **all**

If this argument is not specified, all processes are migrated.

-report

(Optional) Lists the workflow processes whose attachments it would have migrated, but makes no changes.

-batch_size

(Optional) Specifies the number of workflow processes to migrate per batch. If this argument is not specified, it migrates 250 per batch.

-num_procs

(Optional) Specifies the number of child processes to run concurrently if the number of processes is greater than the batch size. If this argument is not specified, it runs 4 child processes concurrently.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following example migrates the VLA-based attachments to GRM relations of all completed workflow processes.

```
migrate_wf_attachments -u=Tc-admin-user -p=password -g=group  
-process=completed
```

migrate_wf_handlers

Transforms the name and/or arguments of workflow handlers from one format to another. Task templates may have one or more workflow handlers associated with them. This utility transforms the handlers associated with templates that are active, as well as template versions that are obsolete but that are still referenced by uncompleted workflow processes or jobs.

This utility uses an XML mapping file to migrate handlers and their arguments. The mapping file top-level nodes define an action (rule or transform) to perform, such as **Replace**, **Remove**, or **Update**, on the individual handler specified within them.

Note:

If you are migrating (as opposed to upgrading) workflows from one Teamcenter environment to another, import the workflows before you run this utility.

SYNTAX

migrate_wf_handlers [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group-name*] [-report=*report-file-name*] [-dryrun] [-listonly] [-templates=*template-to-migrate*, ...] [-templates_file=*path-and-name-of-csv-file*] [-mapping_file=*path-and-name-of-xml-mapping-file*] [-v] [-h]

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the -u and -p arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

If used without a value or if neither the -pf nor the -p argument is used, the system displays an error and asks for a username and password interactively.

This argument is mutually exclusive with the -pf argument.

-pf

Specifies the password file.

If used without a value or if neither the **-pf** nor the **-p** argument is used, the system displays an error and asks for a username and password interactively.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-report

Specifies the output file. If this argument is not specified, the output is written to the standard output.

-dryrun

Runs the command without making any changes. Actions that would have been performed are displayed instead.

-listonly

Lists the handlers and arguments it would have migrated, but makes no changes.

-templates

Limits the migration of handlers to ones owned by the specified templates. If you use more than one template name, separate them with a comma. If a template name has spaces in it, quotes are required around the name. Mutually exclusive with the **-templates_file** argument.

-templates_file

Limits the migration of handlers to ones owned by the templates specified in the named file. The file contains a comma-delimited list of one or more template names. Mutually exclusive with the **-templates** argument.

-mapping_file

Specifies the path and file name of the XML mapping file containing transforms or rules with the old and new names for handlers and arguments. The mapping file provided by Siemens Digital Industries Software to convert handler names and arguments from versions prior to Teamcenter 10.1 is `TC_DATA\wf_handler_migration.xml`.

-v

Displays verbose output and its use is recommended if you are running this utility manually. If this argument is not specified, nothing is displayed when the utility is run.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- After replacing one handler with two or more new handlers using the **Replace** rule in the map file, the new handlers are not available for further processing during the same run of the utility in subsequent mapping file elements. However, the handlers would be available for processing when you run the utility again.
- The **Replace** rule cannot replace an action handler with a rule handler and vice versa.

EXAMPLES

- The following example does not migrate handlers, it only performs a dry run in verbose mode, with the output information sent to the standard output. The `TC_DATA` variable is specified for Windows systems. Because `-u` and `-p` are not specified, the user's operating system credentials are used.

```
migrate_wf_handlers -v -dryrun -mapping_file=
%TC_DATA%\wf_handler_migration.xml
```

- The following example migrates the handlers using the **tcadmin** user with verbose output to the standard output. The `TC_DATA` variable is specified for Linux systems.

```
migrate_wf_handlers -v -u=tcadmin -p=password -mapping_file=
$TC_DATA/wf_handler_migration.xml
```

- The following example migrates the handlers using the **tcadmin** user and a password file with verbose output to the specified report file (**rpt_file**):

```
migrate_wf_handlers -v -u=tcadmin -pf=pswd_file -report=rpt_file
-mapping_file=./map.xml
```

- The following example migrates in silent mode only the handlers used by the specified workflow template:

```
migrate_wf_handlers -templates="Authorization WF Template"
-mapping_file=D:\Temp\MyMigrationMappingFile.xml
```

purge_processes

Purges completed processes based on the last-modified date of the process. Objects such as e-mail messages that reference the process are not deleted. Use the **-f** argument to delete both in-process and completed processes and sever the references between objects and the processes.

Note:

As long as the **-r** argument is not configured then **purge_processes** will automatically purge existing orphan status objects.

SYNTAX

purge_processes

```
[-u=user-id {-p=password | -pf=password-file} -g=group-name] {-d=MM-DD-YYYY| -folder=folder-name} [-force] [-r] [-h] [-pso][pso_batch_size=<batch size>][-pdm][-pdm_batch_size=<batch size>] [-pdm_start_date=MM-DD-YYYY][-pdm_end_date=MM-DD-YYYY]
```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-d

Specifies the cut-off date for processes to be purged. This is the last-modified date of the process. All processes with a last-modified date equal to or before the specified date are purged. The utility will only accept either the **-d** argument or the **-folder** argument, but not both.

If the **-d** argument is configured then do not configure the **-folder** argument.

-folder

Specifies the folder containing the processes to be purged. The folder must exist directly inside the executing user's **Home** folder. The utility will only accept either the **-folder** argument or the **-d** argument but not both.

If the **-folder** argument is configured then do not configure the **-d** argument.

-pso

Purges status objects (**Fnd0ObjectStatus**) associated with completed jobs. Workflow utilizes status objects to store error information so it can be retained until the issue causing the error is resolved. If the **-pso** argument is configured, then after the utility purges processes, it will then purge status objects from any remaining completed processes.

-pso_batch_size

The batch size provided is an integer that indicates the maximum number of status objects (**Fnd0ObjectStatus**) that will get purged. This argument is only valid when the **-pso** argument is configured. If **-pso_batch_size** is not configured when **-pso** is configured, then all status objects (**Fnd0ObjectStatus**) associated with completed jobs will be purged.

-pdm

Purges unreferenced detailed message objects (**Fnd0DetailedMessage**). If **-pdm** is configured and **-pdm_batch_size** is not configured, then then all unreferenced detailed message objects (**Fnd0DetailedMessage**) will be purged.

-pdm_batch_size

The batch size provided is an integer that indicates the maximum number of detailed message objects (**Fnd0DetailedMessage**) that will get purged. This argument is only valid when the **-pdm** option is configured. If **-pdm_batch_size** is not configured when **-pdm** is configured, then all unreferenced detailed message objects (**Fnd0DetailedMessage**) will be purged.

-pdm_start_date

This is the start date in a date range. This argument is only valid when both the **-pdm** and **-pdm_end_date** arguments are also configured. The combination of **-pdm_start_date** and **-pdm_end_date** will be used to form a date range.

-pdm_end_date

This is the end date in a date range. This argument is only valid when both the `-pdm` and `-pdm_start_date` arguments are also configured. The combination of `-pdm_start_date` and `-pdm_end_date` will be used to form a date range.

-f

Deletes in-process and completed processes specified by the `-d` argument from the system and severs references between objects and the processes being deleted. If this argument is not specified, only completed processes that have no referenced objects are purged.

-r

Generates a report of the number and names of processes to be purged without purging the processes.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

The utility will only accept either the `-d` argument or the `-folder` argument but not both. Configure only one of these arguments.

EXAMPLES

To remove all processes that have been modified on or before 15th April 1998, enter the following command on a single line:

```
purge_processes -u=user-id -p=password -g=dba -d=04-15-1998
```

To purge processes as described in the previous example and also purge status objects, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -d=04-15-1998 -pso
```

To remove processes located in the folder named **my_folder** where the folder is located beneath the executing user's **Home** folder, search for processes, type **Job**, using any search criteria desired, and copy the processes into the folder **my_folder**. Execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder
```

To purge processes as described in the previous example and purge status objects, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pso
```

To only purge status objects and not affect processes, create a folder beneath the executing user's **Home** folder and leave the folder empty. For instance, create a folder named **my_folder** and leave it empty. No processes will be affected since the folder is empty, but the **-pso** option will still find and purge status objects associated with completed processes. Execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pso
```

The following examples will all make use of the empty folder named **my_folder** as previously described.

To purge all status objects (FndObjectStatus) associated with completed workflow processes (jobs), execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pso
```

To purge a maximum of 100 status objects associated with completed workflow processes (jobs), execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pso -  
pso_batch_size=100
```

To purge all unreferenced detailed message objects, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pdm
```

To purge a maximum of 200 unreferenced detailed message objects, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pdm -  
pdm_batch_size=200
```

To purge all unreferenced detailed message objects between June 15, 2019 and August 10 2020, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pdm  
-pdm_start_date=06-15-2019 -pdm_end_date=08-10-2020
```

To purge a maximum of 175 unreferenced detailed message objects between June 15, 2019 and August 10, 2020, execute the following command:

```
purge_processes -u=user-id -p=password -g=dba -folder=my_folder -pdm  
-pdm_start_date=06-15-2019 -pdm_end_date=08-10-2020 -pdm_batch_size=175
```

release_man

Releases objects in batch mode without creating workflow processes or audit files.

SYNTAX

```
release_man [-u=user-ID {-p=password | -pf=password-file} -g=dba] [-spec] [-unrelease]
-retain_release_date [-status=status-type] -folder=folder-name [-dataset=dataset] [item=item-ID]
-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]] [-rev=revision-ID] [-relation=relation]
[-datasetName=dataset-name] [-datasetType=dataset-type] [-force] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group. Must be the **dba** group.

See restriction #1.

-spec

Indicates that specifications and BOM view revisions of an item revision in the release folder are released along with the item revision.

-unrelease

Removes the specified status type. See restrictions #2 and #5.

-retain_release_date

Specifies that if the object to be released is already released, the original release date is retained. See restriction #5.

-status

Specifies the status type to be applied to all objects.

See restriction #2.

-folder

Specifies the name of the release folder. Place the objects whose status you want changed into the release folder. See restriction #3.

Status is changed only on the objects one level down within the folder. For example, if you place an item in the folder that contains multiple item revisions, and these item revisions contain multiple datasets, only the status of the item is changed. The status of the item revisions and their datasets is not changed. To change the status of the item revisions, each item revision must be individually placed in the folder at the same level as the item.

Use the **-spec** argument to change the status of specifications and BOM view revisions of item revisions.

-dataset

Specifies that a dataset is released.

-item

Specifies the item ID of the dataset to be released. Use with the **-dataset** argument.

-key

Specifies the key of the dataset to be released. Use with the **-dataset** argument.

Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-rev

Specifies the revision ID of the dataset to be released. Use with the **-dataset** argument.

-relation

Specifies the relation of the dataset to be released. Use with the **-dataset** argument.

-datasetName

Specifies the name of the dataset to be released. Use with the **-dataset** argument.

-datasetType

Specifies the type of dataset to be released. Use with the **-dataset** argument.

-force

Forces the specified release status to be unreleased, even if there is no release type associated with the status. Must be used with the **-unrelease** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. The user must be a member of the **dba** group.
2. The status type must be a valid status type defined for your site.
3. The release folder must be a single folder directly inside the executing user's workspace **Home** folder.
4. The **release_man** utility does not release invalid objects or objects locked by other processes.
5. The **-retain_release_date** and **-unrelease** arguments cannot to be used together.

EXAMPLES

- To apply the **Released** status type to all objects in the **my_folder** folder (including ItemRevision specifications and BOM view revisions), enter the following command on a single line:

```
$TC_ROOT/bin/release_man -u=user-id -p=password -g=dba -spec
-status=Released -folder=my_folder
```

- To remove the **Released** status type from all objects in the **my_folder** folder (including ItemRevision specifications and BOM view revisions), enter the following command on a single line:

```
TC_ROOT/bin/release_man -u=user-id -p=password -g=dba -spec -unrelease
-status=Released -folder=my_folder
```

released_parts_collector

Socket-based server application that receives requests from the **RDV-add-released-parts-queue** workflow handler using the Teamcenter server. Upon receiving the request, it opens the file that contains the list of parts to be processed, locks the file, appends the newly released IA information to the file, and unlocks the file. This application can be used in two modes. When using the **-S** option, it works as a server, and when using the **-C** option, it works as a client application.

SYNTAX

Server mode:

released_parts_collector_server S *host-name:port-number master-file*

Client mode:

released_parts_collector_server C *host-name:port-number*

Command 1: **Cf** *host*

-name:port-number client-IA-file

Command 2: **C** *host-name:port-number stop*

Command 3: **C** *host-name:port-number empty-master-list*
[-h]

ARGUMENTS

-S

Specifies that the application functions as a server.

host-name:port-number

Indicates the port number at which this server should listen for incoming requests.

master-file

Specifies the absolute path to the master XML file to be created/updated with the released parts information. The released parts information is received from either the **RDV-add-released-parts-queue** workflow handler or the **released_parts_collector** command line utility in client mode.

-C

Specifies that the application functions as a client.

host-name:port-number

Specifies the name and port number of the server. This is required when running the utility in client mode.

client-IA-file

Defines the absolute path of an XML file containing the list of released parts. This option must be specified only when running the utility in client mode. When this option is used, the client

application sends the list of parts in the specified file to the server, which appends the new parts to the master list.

stop

Sends a message to stop the server.

empty_master_list

Sends a message to the server to empty the master released parts list.

-h

Displays help for this utility.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

- To start the server on the **trysun12** machine at port **5567** and create the file **master_list.xml** file, enter the following command on a single line:

```
released_parts_collector -S trysun12:5567 /tmp/master_list.xml
```

- To send the contents of the file **/tmp/my_released_list.xml** to the server running on **trysun12** at port **5567** and add the new list of parts to the master list of released parts, enter the following command on a single line:

```
released_parts_collector -Cf trysun12:5567  
/tmp/my_released_list.xml
```

- To stop the server running on **trysun12** at port **5567**, enter the following command on a single line:

```
released_parts_collector -C trysun12:5567 stop
```

- To empty the released parts list XML file on the server running on **trysun12** at port **5567**, enter the following command on a single line:

```
released_parts_collector -C trysun12:5567 clear_master_file
```


tc_workflow_postprocess

Executes a specific action on a specific task in the workflow process from which the related background process was initiated.

For example, you can use this utility to evoke the **complete** action on the **Review** task in the workflow process from which a background tessellation process was initiated. This utility then prompts the workflow task to either execute an action (defined with the **-action** argument) or submit a decision (defined with the **-signoff** argument). This can be useful when the conditions to execute the action are not met until the background process completes. In these cases, the user has typically moved on to other tasks or ended the session.

Use the **-member_group** and **-member_role** arguments to define the group/role used for the background process. This is useful at sites where users have multiple groups/roles and the user has changed to a group/role that is different from his default login group/role while initiating the background process. These arguments allow the **tc_workflow_postprocess** utility to assume the same group/role the user was using at the time the workflow process was initiated. It is expected that the same group/role is required to execute any action in that workflow process on behalf of the user.

SYNTAX

```
tc_workflow_postprocess [-u=user-id {-p=password | -pf=password-file}
-g=group-name] -status_xfer_type=transfer-type
[itemid=item-id | -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
-revid=rev-id -dsname=dataset-name -dataset_tag=dataset-tag -task_tag=tag
[-member_group=group] [-member_role=role] [-action=action-name] [-
trigger_comment=comment] [-signoff=decision] [-signoff_comment=comment] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

If the utility is called using the **EPM-invoke-system-action** or **EPM-invoke-system-rule** handlers, use **-u=-** and **-p=-** arguments with the utility so it inherits the session authentication.

-status_xfer_type

Indicates the type of status transfer to perform. An argument value of **cae_mesh** transfers the release status to an associated **CAEMesh** dataset. Any other value, or not supplying the argument, transfers the release status to an associated **DirectModel** dataset. This argument and value are valid only if the **-dataset_tag** argument is specified; otherwise, this argument is ignored.

-itemid

Identifies the item under which to locate the **CAEMesh** dataset. The utility transfers the release status to the **CAEMesh** dataset at the location indicated by this argument, the **-revid**, and the **-dsname** arguments. This argument and value must be supplied only if the argument/value **-status_xfer_type=cae_mesh** is specified; otherwise, it is ignored.

-key

Specifies the key of the item under which to locate the **CAEMesh** dataset. The utility transfers the release status to the **CAEMesh** dataset at the location indicated by this argument, the **-revid**, and the **-dsname** arguments. This argument and value must be supplied only if the argument/value **-status_xfer_type=cae_mesh** is specified; otherwise, it is ignored.

Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *Introduction to multifield keys* in *BMIDE for Data Model Design*.

-revid

Identifies the item revision under which to locate the **CAEMesh** dataset. The utility transfers the release status to the **CAEMesh** dataset at the location indicated by this argument, the **-itemid**, and the **-dsname** arguments. This argument and value must be supplied only if the argument/value **-status_xfer_type=cae_mesh** is specified; otherwise, it is ignored.

-dsname

Identifies the name of the **CAEMesh** dataset to which to transfer the release status. The utility transfers the release status to the **CAEMesh** dataset at the location indicated by this argument, the **-itemid** and the **-revid** arguments. This argument and value must be supplied only if the argument/value **-status_xfer_type=cae_mesh** is specified; otherwise, it is ignored.

-dataset_tag

Specifies the tag of the dataset to which the release status is transferred. The release status of the primary object (target object of the workflow process) is applied to the specified dataset.

-task_tag

Provides a text representation of the tag of the workflow task. The value can be extracted from the XML file provided by either the **EPM-invoke-system-action** or **EPM-invoke-system-rule** handler.

-member_group

Assumes the defined group name before executing an action on the workflow process.

Use when users have multiple groups/roles and the user is expected to change to a group different from their default login group while initiating the background process.

This argument allows the utility to assume the same group the user was using at the time the workflow process was initiated. It is expected that the same group is required to execute any action in that workflow process on behalf of the user.

-member_role

Assumes the defined role name before executing an action on the workflow process.

Use when users have multiple groups/roles and the user is expected to change to a role different from their default login role while initiating the background process.

This argument allows the utility to assume the same role the user was using at the time the workflow process was initiated. It is expected that the same role is required to execute any action in that workflow process on behalf of the user.

-action

Defines which action to trigger in the workflow task specified with the **-task_tag=tag** argument.

Valid actions are: **assign**, **start**, **complete**, **skip**, **suspend**, **resume**, **undo**, **abort**, and **demote**. These action values are not case sensitive.

-trigger_comment

Provides the comment when triggering the action specified in the **-action=action-name** argument.

-signoff

Specifies the utility will perform a signoff with the specified decision.

Valid signoff values are: **approve**, **reject**, **nodecision**. These signoff values are not case sensitive.

-signoff_comment

Provides the comment for the signoff specified with the **-signoff=decision** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following example checks the group/role defined by the **-member_group** and **-member_role** arguments (**Body/Designer**) against the group/role the user is currently logged on with. If they do not match, the group/role of the background processes group/role is changed to **Body/Designer**.

The **complete** action is then invoked for the task specified by the **-task_tag** argument and the comment **Tessellation completed successfully** is displayed.

```
$TC_ROOT/bin/tc_workflow_postprocess
  -member_group=Body -member_role=Designer
  -task_tag=AZwszeaegnHvqDAAAAAAAAAAAAAA -action=Complete
  -trigger_comment="Tessellation completed successfully"
```

- The following example checks the group/role defined by the **-member_group** and **-member_role** arguments (**Body/Designer**) against the group/role the user is currently logged on with. If they do not match, the group/role of the background processes group/role is changed to **Body/Designer**.

The signoff decision **No Decision** is then made for the task specified by the **-task_tag** argument.

```
$TC_ROOT/bin/tc_workflow_postprocess
  -member_group=Body -member_role=Designer
  -task_tag=AZwszeaegnHvqDAAAAAAAAAAAAAA -signoff=NoDecision
  -signoff_comment="Tessellation failed - disk full"
```

- The following example illustrates the use of the **-dataset_tag** argument to apply the release status of a rendering parent object to the related child object.

```
$TC_ROOT/bin/tc_workflow_postprocess
  -dataset_tag=AXwszeagnHvqDAAAAAAAAAAAAAA
```

- The following example transfers the release status from a **UGMASTER** dataset to its corresponding **CAEMesh** dataset:

```
$TC_ROOT/bin/tc_workflow_postprocess  
-dataset_tag=QZPBK4_6x4$kbDAAAAAAAAAAAAA  
-status_xfer_type=cae_mesh -itemid=000266  
-revid=A -dsname=000266/A
```

verify_tasks

Finds all corrupted Workflow tasks, jobs, and other associated internal task model objects in order to delete them from the database. If a corrupted object, such as a job, is referenced in a folder, the reference is removed and the job is deleted.

SYNTAX

```
verify_tasks [-u=user-id {-p=password | -pf=password-file} -g=group-name] [-m={list | delete}] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-m

Sets mode to one of the following:

list

Lists corrupted jobs and tasks without deleting them.

delete

Lists and deletes corrupted jobs and tasks.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.



6. Indexing and search utilities

runTcFTSIndexer

Indexes data into the Solr indexing engine. Run this command from the *FTS_INDEXER_HOME* directory, for example, *TC_ROOT\TcFTSIndexer\bin*. This command is not required to be run in a specific environment.

SYNTAX

runTcFTSIndexer -debug -maxconnections -status -stop -service -shutdown -task=[objdata | multisite | structure | fourgd]:flow-action -h

Note:

Operating system considerations:

Linux	A <i>.sh</i> extension is required, for example, <i>./runTcFTSIndexer.sh</i> . If a UID contains a special character, you must enclose it in straight, single quotation marks. UIDs can contain A-Z , a-z , 0-9 , \$, and _ .
Windows	The <i>.bat</i> extension is not required for runTcFTSIndexer . For UIDs on Windows, do not enclose in straight, single quotation marks.

ARGUMENTS

-debug

Reports a summary of the flow in progress, including connections and the logs associated with them, to the command window and to the **TcFTSIndexer** logs in **TcFTSIndexer\logs**. Run **-debug** in a separate command window.

```
runTcFTSIndexer -debug
```

-maxconnections

Sets a new value for maximum **tcserver** connections to use at any given time, for example:

```
runTcFTSIndexer -maxconnections=5
```

-status

Checks the status of the indexer and reports the flows running in the indexer. For example, the following shows the status for all the flows:

```
runTcFTSIndexer -status
```

This argument can also be used with the **-task** argument. For example, the following command shows the status of the **objdata:index** flow:

```
runTcFTSIndexer -status -task=objdata:index
```

-stop

Stops indexing flows that use an interval.

```
runTcFTSIndexer -stop
```

This argument can also be used with the **-task** argument. For example, the following command stops the **objdata:sync** flow with intervals:

```
runTcFTSIndexer -stop -task=objdata:sync
```

-service

Starts the indexer as a Java Remote Method Invocation (RMI) service in a console, for example:

```
runTcFTSIndexer -service
```

This argument can also be used with the **-task** argument to run a flow when starting the service, for example:

```
runTcFTSIndexer -service -task=objdata:index
```

-shutdown

Shuts down the service after stopping all the flows.

```
runTcFTSIndexer -shutdown
```

-task

Runs an indexing task in this format:

```
-task=type:flow-action
```

Flow actions are specific to the type of task.

objdata	Indexing actions to support object data indexing
admin	Administrative actions to support clone and merge indexing
fourgd	Indexing actions to support 4GD
multisite	Indexing actions to support multi-site published records
structure	Indexing actions to support structured content indexing

-h

Displays the help for this utility.

OBJDATA FLOW ACTIONS

- **objdata:build_suggester**

Rebuilds the suggestions dictionary for Solr. This flow is separate from the sync flow. Run **objdata:build_suggester** if the **objdatasyncstep.build_suggester_inline** property in the **TcFtsIndexer_objdata.properties** file is set to **false**.

- This action can use the **-interval=seconds** argument, for example, to rebuild the suggestions dictionary every five minutes (300 seconds):

```
runTcFtsIndexer -task=objdata:build_suggester -interval=300
```

The interval value must be 300 or greater. Running this flow action with an interval is recommended.

- You can rebuild the suggestions dictionary once, for example:

```
runTcFtsIndexer -task=objdata:build_suggester
```

- **objdata:clear**

Clears existing indexed data, records, and cached files. This option is most often used prior to running **objdata:index**. When you run **-task=objdata:clear**, specify one of these options when you are prompted:

1 clears the indexer cache.

2 clears the Solr index.

3 clears the object data indexing records from the Accountability table.

4 clears the data covered by options 1, 2, and 3.

5 clears the UIDs passed in from the Accountability table.

When you specify **5**, provide the full path to the text file containing the UIDs.

6 cancels the clear flow.

7 cleans up the scratch table records, where the last saved date precedes the last processed date of the subscription table.

If you run **-task=objdata:clear** in an environment that is set to **Clone Ready**, it fails in order to prevent accidentally clearing the index.

Example:

To clear a list of UIDs, run the command:

```
runTcFTSIndexer -task=objdata:clear
```

At the prompt:

```
Please provide a clear option from above list [1-7]:
```

Enter 5. The prompt returns:

```
Provide full path to input uid text file.
```

Enter the full path and file name:

```
c:\my_uid_directory\uids.txt
```

- **objdata:embed**

Embeds the indexed file contents included in the configured knowledge sources for use in Teamcenter Artificial Intelligence (AI) Chat.

- **objdata:embedschema**

Embeds the indexed object data for use in Natural Language Search.

- **objdata:errors**

Returns the errors for indexing failures to a specified directory. The directory must be empty. You can optionally specify the number of errors to be returned. The default is 50 errors.

The error directory contains a directory for each UID of the failed object. Each UID directory contains the following information:

- Properties such as object name and type provide information on the failed objects.
- The TC XML and Solr XML files are generated and saved for debugging.
- Syslog information.
- **TcFTSIndexer** log files with the errors.

In this example, the first 100 errors are sent to the specified output directory:

```
runTcFTSIndexer -task=objdata:errors d:\TcFTSIndexer\errors 100
```

- **objdata:index**

Indexes data for the time period specified in

`TC_ROOT\TcFTSIndexer\conf\TcFTSIndexer_objdata.properties` without clearing the existing index.

For a clean start, first use **objdata:clear** to clear indexed data and cached files. However, if you run it in a **Clone Ready** environment, it fails in order to prevent accidentally clearing the index.

- **objdata:indexclone**

Indexes the specified time period without clearing the existing index. The start time is set to when the **Clone Ready** state was set. The end time is set in the `confobjdata.properties` file.

This indexing flow supports the clone and merge indexing approach, which is a method to manage upgrades and patches more efficiently by reindexing a cloned environment and then merging it to an updated production environment.

The **objdata:indexclone** task picks up the delta of indexing changes in the production environment that occurred while the cloned environment was being indexed. This flow depends on the **administrative flow actions**.

- **objdata:indexlargetload**

Queries, stages, and indexes data in chunks for the time period specified in

`TC_ROOT\TcFTSIndexer\conf\TcFTSIndexer_objdata.properties` without clearing the existing index.

Use this flow to better handle larger data that may pose challenges for the indexer due to Java heap limitations.

- **objdata:indexsyncfailures**

Indexes synchronization failures that required manual intervention.

- **objdata:recover**

Recovers failed indexed objects.

If failures are reported during initial indexing, run the recover flow action after initial indexing completes. Recover failures from the initial index by running **objdata:recover** repeatedly until there are no failures reported or the failures are consistent and need to be fixed.

You can choose to fix the errors or leave them for later and proceed with the synchronization flow. If you leave them, these errors are logged as failures during synchronization. You can return at a later time to fix them. The recover flow action operates by time slice. The recover flow action processes all failed objects together.

- **objdata:recoverlargeload**

Recovers failed indexed objects.

If failures are reported during **indexlargeload** flow action, run the **objdata:recoverlargeload** flow action after the **indexlargeload** action completes. Recover failures from the **indexlargeload** action by running **objdata:recoverlargeload** repeatedly until there are no failures reported or the failures are consistent and need to be fixed.

You can choose to fix the errors or leave them for later and proceed with the synchronization flow. If you leave them, these errors are logged as failures during synchronization.

- **objdata:show**

Returns objects that are in a specified indexing state in two text files. Specify the state code and a directory for the output files using the form **-task=objdata:show** *n uid_file_dir*.

The **show** number to specify the state is **1**, **2**, or **3**:

1 returns objects in the replication pending state.

2 returns objects in the indexing complete state.

3 returns objects in the indexing failed state.

In this example, the output files contain the objects that failed indexing:

```
runTcFTSIndexer -task=objdata:show 3 d:\TcFTSIndexer\uid_output
```

The output returns two text files:

- **uid.txt**

Contains the UUIDs of the objects that match the specified state. Each UUID is on one line.

You can synchronize the objects again using **objdata:sync** *uid_file_dir*. Specify the directory containing the **uid.txt** file, for example:

```
runTcFTSIndexer.bat -task=objdata:sync d:\TcFTSIndexer\uid_output
```

- **uid_prop.txt**

Contains UUIDs in the output format *UUID | object_string | object_type*, for example:

```
TRa3S5qdSDyB | Breaker Panel Anchor Plate/AP02-A | Physical Part Revision
```

- **objdata:sync**

Updates the index with changes to data between the previous run and the current run.

- The **sync** action can take the **-interval=seconds** argument.

For example, to synchronize object data using the stand-alone indexer every 300 seconds (5 minutes):

```
runTcFTSIndexer -task=objdata:sync -interval=300
```

The value must be greater than **0**.

To run **sync** once, omit **-interval=seconds**.

- The **sync** action can take a *file path* to a **uid.txt** file.

You can synchronize the objects again using **objdata:sync filepath**. Specify the directory containing the **uid.txt** file, for example:

```
runTcFTSIndexer.bat -task=objdata:sync d:\TcFTSIndexer\uid_output
```

- **objdata:test**

Verifies whether the environment is set up correctly prior to running the indexer.

ADMINISTRATIVE FLOW ACTIONS

- **admin:cloneready**

Sets the production environment to **Clone Ready** in preparation for creating a cloned environment. After this action is applied, the cloned environment can be upgraded or patched and then indexed using **objdata:index**.

- **admin:clonevalidate**

Confirms that the indexing information in the upgraded production environment and the upgraded cloned environment are the same.

Run this flow action to determine whether the production environment is ready for the indexing system to be merged.

After the merge, pick up the delta of indexing changes in the upgraded production environment using **objdata:index clone**.

- **admin:clonecomplete**

Removes the **Clone Ready** setting on the indexing system in the upgraded production environment. The environment must have been previously set to **Clone Ready**.

- **admin:status**

Provides the means to verify the current clone status of the production environment.

MULTI-SITE FLOW ACTIONS

- **multisite:clear**

Clear indexed data and cached files for existing multi-site published records.

Use prior to running **multisite:index**.

- **multisite:index**

Without clearing the existing indexed multi-site published records, index for the time period specified in the `TC_ROOT\TcFTSIndexer\conf\TcFTSIndexer_multisite.properties` file. For example:

```
runTcFTSIndexer -task:multisite:index
```

During indexing, duplicate published records are removed.

For a clean start, use **multisite:clear** to clear indexed data for multi-site published records and cached files.

- **multisite:indexuids**

Index the UIDs for a specific Object Directory Services (ODS) site.

Specify the ODS site name and the directory containing the **uid.txt** file, for example:

```
runTcFTSIndexer -task=multisite:indexuids ODS_Site_name D:\UID_dir
```

- **multisite:indexsyncfailures**

Index synchronization failures that required manual intervention.

- **multisite:sync**

Updates the index with changes to published record data between the previous run and the current run.

The **sync** action can take the **-interval=seconds** argument to repeat the **sync** action at the specified interval. For example, to synchronize multi-site published record data every 300 seconds (5 minutes):

```
runTcFTSIndexer -task=multisite:sync -interval=300
```

The value must be greater than **0**.

To run **multisite:sync** once, omit **-interval=seconds**.

- **multisite:recover**

Recovers failed indexed published record objects.

If some published records fail during indexing, run the recover flow action after multi-site indexing completes. Recover failures by running **multisite:recover** repeatedly until there are no failures reported or the failures are consistent and need to be fixed.

- **multisite:test**

Verify whether the environment is set up correctly prior to running the multi-site indexer.

STRUCTURE FLOW ACTIONS

- **structure:recoverfailures**

Changes all product configurations with failed states to the **ReadyToIndex** state or the **MarkedForDeletion** state. For example:

```
runTcFTSIndexer -task=structure:recoverfailures
```

- **structure:reset** *product-configuration-UID*

Resets the given product configuration UID setting to the **ReadyToIndex** state or the **MarkedForDeletion** state. For example:

```
runTcFTSIndexer -task=structure:reset 'goZRkWxoqd$DyB'
```

- **structure:resetall**

Downloads the latest transform and schema files, resets all active product configurations to the **ReadyToIndex** state, and resets all deleted product configurations back to the **MarkedForDeletion** state. For example:

```
runTcFTSIndexer -task=structure:resetall
```

- **structure:show**

Shows a summary of all configured product configurations, for example:

```
runTcFTSIndexer -task=structure:show
```

- **structure:sync**

Queues synchronization and delete actions for all product configurations, for example:

```
runTcFTSIndexer -task=structure:sync
```

- **structure:syncone** *product-configuration-UID*

Queues synchronization and delete actions for a single product configuration UID, for example:

```
runTcFTSIndexer -task=structure:syncone goZRkwxoqd12DyB
```

- **structure:test**

Verifies that the environment is set up correctly prior to running the indexer, for example:

```
runTcFTSIndexer -task=structure:test
```

4GD FLOW ACTIONS

The 4GD product structure is initially indexed using the **bomindex_admin** utility.

- **fourgd:reset** *product-configuration-UID*

Resets the given 4GD product configuration UID setting to the **ReadyToIndex** state. For example:

```
runTcFTSIndexer -task=fourgd:reset goZRkwxoqd$DyB
```

- **fourgd:resetall**

Downloads the latest transform and schema files, resets all active 4GD product configurations to the **ReadyToIndex** state. For example:

```
runTcFTSIndexer -task=fourgd:resetall
```

- **fourgd:show**

Shows a summary of all 4GD configured product configurations, for example:

```
runTcFTSIndexer -task=fourgd:show
```

- **fourgd:sync**

Queues synchronization and delete actions for all 4GD product configurations, for example:

```
runTcFTSIndexer -task=fourgd:sync
```

- **fourgd:syncone** *product-configuration-UID*

Queues synchronization and delete actions for a single 4GD product configuration UID, for example:

```
runTcFTSIndexer -task=fourgd:syncone goZRkWxoqd12DyB
```

```
./runTcFTSIndexer.sh -task=structure:syncone 'Abed$b8dBhGXHA'
```

awindexerutil

Refreshes indexed objects if you make changes to them and you want the synchronization flow to refresh the index for those objects. The **awindexerutil** utility marks those objects to be picked up during the next synchronization flow batch. This allows you to update the index without the downtime of a full index flow.

You can refresh your index for only the delta of changes since the last completed synchronization. You can index changes for types and properties that have been added, modified, or removed, by performing a delta update.

Running **awindexerutil** does not interfere with current synchronization flows.

Run the utility from the *TC_ROOT\bin* directory (*TC_BIN* if it's set).

SYNTAX

awindexerutil -u=user-id -p=password -g=group [-refresh] [-delta [-dryrun] [-daterange]] -h

ARGUMENTS

-u

Specifies the user ID. The user needs administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-g

Specifies the group associated with the user.

-refresh

Starts the synchronization flow according to the batch size.

-delta

Updates the index for the delta of changes to object types and properties since the last completed synchronization. Be sure to run **-delta -dryrun** to evaluate the changes before running the **-delta** index update.

Use this approach to .

-dryrun

Use with **-delta** to compare the changes in object types and properties for delta reindexing. No indexing is performed. The differences are output to the command window and a log file. The *.log* file path is returned after the operation is complete.

-daterange

Use with **-delta** or **-delta -dryrun** to set a date range for object types and properties that can be indexed in the updated schema. To set the date range, use the following year-month-day form:

YYYY/mm/dd HH:MM:SS-YYYY/mm/dd HH:MM:SS

The time (**HH:MM:SS**) is optional. You can specify multiple ranges in a comma-separated list. If the specified date range contains spaces, enclose the entire specification in quotation marks.

-classification

Used together with the **-delta** argument, evaluates the schema files and marks the changes in classification data for indexing in the next scheduled indexing synchronization run. Run the utility with the **-classification -delta -dryrun** arguments to receive a list of changes that will be marked for indexing when the **awindexerutil** is run.

-h

Displays help for this utility.

EXAMPLES

- Start the synchronization flow to refresh data objects:

```
awindexerutil -u=admin -p=pwd -g=group -refresh
```

- Start the dry run comparison of the delta of indexing changes. All instances of new, modified, and deleted types and properties are identified. For newly indexed types, only those instances that were changed during the specified date range (2012/12/12 - 2016/12/31) are included. Only a report is returned. No indexing changes are made.

```
awindexerutil -u=admin -p=pwd -g=group -delta -dryrun
-daterange=2012/12/12-2016/12/31
```

- Start reindexing the delta of changes. Updates all new, modified, and deleted types and properties.

```
awindexerutil -u=admin -p=pwd -g=group -delta
```

- Start reindexing the delta of changes. Updates all new, modified, and deleted types and properties. For types that are newly added, only those objects last modified in the specified date range are indexed.

```
awindexerutil -u=admin -p=pwd -g=group -delta -daterange=  
"2015/12/12 15:30:00 - 2015/12/31 22:00:00, 2016/02/01 07:00:00  
- 2016/12/31 14:00:00"
```

- Mark changes in classification data for indexing:

```
awindexerutil -u=admin -p=pwd -g=group -delta -classification
```

aw_search_config_manager

Manages mapping configurations used when customizing asynchronous file content indexing. Mapping configurations specify how data extracted from non-standard files, such as CAD files, are indexed into Solr.

Run the utility from the `TC_ROOT\bin` directory. (You can also use the `TC_BIN` directory if it is configured.)

SYNTAX

```
aw_search_config_manager -u=user-ID {-p=password | -pf=password-file} -g=group
  [-import -dir= config-directory]
  [-export -config_id= config-id -output= output-file-path]
  [-remove -config_id= config-id]
  [-list]
  [-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted for a user ID and password.

-p

Specifies the user's password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-import

Imports the configuration files from the given directory.

-dir

Used with **-import** to specify the directory from which configuration files are imported.

-export

Imports the configuration files from the given directory.

-output

Used with **-export** to specify the full path, including file name, where you want the JSON configuration saved.

-remove

Removes a mapping configuration.

-config_id

- When used with **-export**, specifies the ID of the configuration to export.
- When used with **-remove**, specifies the ID of the configuration to remove.

-list

Displays a list of the stored configuration IDs.

-h

Displays help for this utility.

EXAMPLES

For example, a mapping configuration file, *C:\mappingConfigs\extConfig.json*, contains a mapping configuration with the ID *extConfig*. This ID (*extConfig*) contains rules that allow the file content transform process to transform data extracted from external sources to a format that can be indexed and searched.

Run the following command to upload all configurations present in the *C:\mappingConfigs* directory:

```
aw_search_config_manager -u=admin -p=pwd -g=dba -import -dir=c:\mappingConfigs
```

Configurations present within *extConfig.json* are now available for use in the transform process.

- List all available configurations:

```
aw_search_config_manager -u=admin -p=pwd -g=dba -list
```

- Export a configuration to a JSON file:

```
aw_search_config_manager -u=admin -p=pwd -g=dba -export -config_id=extConfig  
-output=c:\exportedConfigs\extConfig.json
```

- Remove an existing configuration:

```
aw_search_config_manager -u=admin -p=pwd -g=dba -remove -config_id=extConfig
```

solrCloudSetUp

If you installed or upgraded to SolrCloud using Deployment Center, you can migrate to SolrCloud using the SolrCloud utility. Running this utility sets up ZooKeeper, creates configsets and collections, sets up basic authentication, and copies your data so that a full index is not required.

Syntax

solr-version.solrCloudSetUp.bat -migrate -u=user name -p=password -copy_data

Arguments

-dry_run Checks the existing collections in SolrCloud.

Example:

```
solr-version.solrCloudSetUp.bat -dry_run
```

-migrate Migrates the existing standalone Solr setup, including ZooKeeper, configsets and collections, and basic authentication. Use with the **-copy_data** argument to copy all existing data to SolrCloud.

Example:

```
solr-version.solrCloudSetUp.bat -migrate -u=user name  
-p=password -copy_data
```

-upload_config Uploads new or existing solr config to ZooKeeper.

Example:

```
solr-version.solrCloudSetUp.bat -upload_config -d=directory  
-n=config name
```

-create_new_collection Creates a new collection with an existing Solr config uploaded in ZooKeeper.

Example:

```
solr-version.solrCloudSetUp.bat -create_new_collection  
-n=config name -c=collection name -u=user name -p=password
```

-create_new_collection_with_config Creates a new collection with a new config.

Example:

```
solr-version.solrCloudSetUp.bat
-create_new_collection_with_config -d=directory -n=config name
-c=collection name -u=user name -p=password
```

-update_config Updates all existing standalone Solr collections and config to ZooKeeper.

Example:

```
solr-version.solrCloudSetUp.bat -update_config
```

-u= Specifies the user name.

-p= Specifies the user password.

-d= Specifies the collection config directory.

-n= Specifies the config name.

-c= Specifies the new collection name.



7. Data sharing utilities

admin_data_export

Exports administration data to a file. You can use the file to import the data to another site, such as from a test or upgrade environment into a production environment. You can also use the file to compare data between two sites. You can alternatively perform full or partial export of administration data using TEM.

SYNTAX

```
admin_data_export -u=user-ID {-p=password | -pf=password-file} -g=group
{-adminDataTypes=data-type1, data-type2,...,data-typeN | all}
{-outputPackage=absolute-path-file-name}
[-inputCriteria= <className>{attr1=value1,attr2=value2,...,attrN=valueN}]
[-listTypes]
[-session_options=option1:value1,option2:value2:...]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If this argument is used without a value, the operating system user name is used.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-adminDataTypes

Specifies a comma-separated list of administration data types you want to export. You can also specify the **all** value to export all the administration data types from the local site.

-outputPackage

Specifies the fully qualified path to the location for the generated export file.

-inputCriteria

Specifies the criteria used to generate an export file containing a partial amount of the site's administration data. You must specify the class name of the administration data first, followed by the attributes and their values that identify the data you want to export.

If you do not specify this argument, the utility does a full export of all administration data at the site. Following are some examples of input criteria. The attributes and values listed here may not match your Teamcenter data.

Admin data type	Selection by	Class name	Example attribute
AccessManager	Access Control List	AM_ACL	{ACL_Name=value}
LogicalObjects	Logical Object Name	LogicalObjectsByName	{type_name=value1,type_name=value2}
Organization	User	User	{user_name=value}
	Group	Group	{name=value}
Projects	Projects	TC_Project	{project_id=value}
Preferences	Preference Name	PreferenceByName	{preference_name=value}
	Group	PreferencesByGroup	{group_name=value}
	Role	PreferencesByRole	{role_name=value}
	User	PreferencesByUser	{user_name=value}
	Category	PreferencesByCategory	{category_name=value}
RevisionRules	Rule	RevisionRules	{object_name=Latest*}
SavedQueries	Query Name	ImanQuery	{query_name=value}
Stylesheets	Stylesheet Name	StylesheetByDatasetName	{dataset_name=value}
	Preference Name	StylesheetByPreferenceName	{preference_name=value}
Subscriptions	Subscription	ImanSubscription	{fnd0Name=val}

Admin data type	Selection by	Class name	Example attribute
UnitDefinitions	ID	UnitDefinitionsById	{unit_definition_id=value}
WorkflowTemplates	Template Name	WorkflowTemplatesByName	{template_name=value}

-listTypes

Displays a list of administration data types that can be exported.

-session_options

Indicates session options that you want the utility to use for the current export. Accepts valid session option-value pairs in a comma delimited format. The option name and value are delimited with a colon.

Caution:

If you do not supply this argument, the **opt_traverse_ref_org** option value is set to **True** and the utility skips user information updates.

You must export or import organization objects (such as users and groups) before importing other objects. Otherwise, the import of other objects fails.

If you supply the **opt_traverse_ref_org** option with the value set to **False**, the utility updates user object data. This may cause the user password to change at the importing site.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment.

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

This utility does not function if the ownership of 4GD replica objects is transferred to the target site. It must be run before any ownership transfers take place.

EXAMPLES

- To export all organization, preferences, and projects administration data from a site:


```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -adminDataTypes=Organization,Preferences,Projects
                  -outputPackage=c:\temp\admin_data\siteA\siteA.zip
```

- To list the available administration data types:

```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -listTypes
```

- To do a partial export of only Access Manager rules named **System**:

```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -adminDataTypes=AccessManager
                  -inputCriteria=AM_ACL{ACL_Name=System}
                  -outputPackage=c:\temp\admin_data\export_partial_am_system.zip
```

- To do a partial export of only Workflow Templates named **TCM Release Process**:

```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -adminDataTypes=WorkflowTemplates
                  -inputCriteria="WorkflowTemplatesByName{template_name=TCM Release Process}"
                  -outputPackage=c:\temp\admin_data\export_partial_wftemplate_release.zip
```

- To export Access Manager rules with referenced organization data:

```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -adminDataTypes=AccessManager
                  -session_options=opt_traverse_ref_org:false
                  -outputPackage=c:\temp\admin_data\export_partial_am_system.zip
```

- To perform a partial export of organization data, pass the input criteria and session options, with the **-opt_traverse_ref_org** option value set to **False**.

- **Example 1**

To export the user whose user name is **PLMUser**:

```
admin_data_export -u=admin-username -p=admin-password -g=dba
                  -adminDataTypes=Organization
                  -outputPackage=C:\Temp\OrganizationPR_User.zip
                  -inputCriteria=User{user_name=PLMUser}
                  -session_options=opt_traverse_ref_org:false
```

- **Example 2**

To export the group having the name **PLM**:

```
admin_data_export -u=admin-username -p=admin-password -g=dba  
-adminDataTypes=Organization  
-outputPackage=C:\Temp\OrganizationPR_Group.zip  
-inputCriteria=Group{name=PLM}  
-session_options=opt_traverse_ref_org:false
```

admin_data_import

Imports administration data from the specified input package into the current Teamcenter environment. (To create a site package file, export administration data.)

A conflict occurs when the data exists at both the exporting and importing sites but does not contain the same values at both. For each administration data type, you can specify how conflicting data is handled during import.

The utility allows you to generate an import report without executing the actual import, referred to as a dry run. You can use the import report (dry run) to analyze how the input package potentially impacts the current environment. For an actual import, the utility adds a report to an import history report that contains a record of all the previous imports performed at the current environment. You can use this report to analyze how administrative data has changed over time at the importing site due to prior imports. Import reports (dry run and actual) are located in the `TC_ROOT\logs\import_history` directory.

Caution:

If you have an input file with Access Manager data from older Teamcenter releases (earlier than Teamcenter 13.1), ensure that the input file is regenerated with a minimum Teamcenter 13.1 version before importing.

SYNTAX

```
admin_data_import -u=user-ID {-p=password | -pf=password-file} -g=group
-inputPackage=input-file-name-and-path
-adminDataTypes=admin-data-type1,admin-data-type2, ... | all
-mergeOption=admin-data-type1,merge-option1:admin-data-type2,merge-option2, ...
[-dryrun] [-skipPackageValidation] [-listTypes] [-listMergeOptions]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-inputPackage

Specifies the fully qualified path to the location for the file to import.

-adminDataTypes

Specifies a comma separated list of administration data types you want to import. You can also specify the **all** value to import all data types in the input package.

Tip:

Use the **-listTypes** argument to display a list of allowable administration data types.

-mergeOption

Indicates how the utility handles data conflicts (merge options) between the target (importing) site administration data and the source (exporting) site data in the input package.

Tip:

Use the **-listMergeOptions** argument to display a list of allowable merge options for each administration data type the import file contains.

-dryrun

Generates an import history report that shows the impact on the importing site's administration data without actually importing the data.

-skipPackageValidation

Specifies the import file is not validated for proper content prior to importing the package.

If you do not specify this argument, the utility ensures the data in the import file is valid and fails the import if it is not.

-listTypes

Displays the supported administration data types that can be imported from the input package.

-listMergeOptions

Lists the supported merge options configured for the available administration data types in the import file. You must specify the **-inputPackage** argument to point to the import file.

The merge options are:

override_with_source

Replaces the target site administration data with the source site administration data.

keep_target

Keeps the existing administration data at the importing site when there are merge conflicts.

choose_latest

Replaces the conflicting administration data with the latest administration data based on time stamp.

choose_target_for_unresolvable_conflicts

Keeps the existing administration data at the importing site for nontrivial or unresolvable merge conflicts. For all other merge conflicts, uses the latest administration data based on time stamp.

session_options

Indicates session options that you want the utility to use for the current command. Accepts valid session options in a comma delimited format.

If you supply the **opt_traverse_ref_org** in this argument value, the utility traverses the organization data. Otherwise its value is passed as true (default behavior).

Caution:

You must export/import organization objects (such as users and groups) before importing other objects. Otherwise the import of other objects fails.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment.

FILES

As specified in Log files produced by Teamcenter.

EXAMPLES

- Imports administration data from the specified input package into the current Teamcenter environment:

```
admin_data_import -u=admin-username -p=admin-password -g=dba
-adminDataTypes=all -inputPackage=C:\temp\admin_data\siteB\siteB.zip
```

- Performs a dry run import from the specified input package with merge options:

```
admin_data_import -u=admin-username -p=admin-password -g=dba  
-dryrun -adminDataTypes=all  
-inputPackage=C:\temp\admin_data\siteB\siteB.zip  
-mergeOption=AccessManager:override_with_source,Organization:  
override_with_source,Preferences:choose_latest,  
RevisionRules:choose_target_for_unresolvable_conflicts,SavedQueries:keep_target,  
Stylesheets:keep_target,  
Subscriptions:keep_target,WorkflowTemplates:override_with_source
```

ar_recover

Attempts to perform a recovery and cleanup of partially created data during Multi-Site Collaboration archive and restore operations when IDSM an ODS connection failures have occurred.

During the archive and restore critical operations, a recovery text dataset containing the operation ID and object UUIDs is created in the current logged-in user's **NewStuff** folder. After a successful operation, the recovery dataset is automatically deleted. However, if a critical failure occurs, the dataset is not deleted and you can use the **ar_recover** utility to attempt recovery of the data.

SYNTAX

```
ar_recover -u=user-ID {-p= password | -pf=password-file} -g=group
-f={recover | list | update} [-ds_name=name][-extinct_siteID=id] [-new_siteID=id][-verbose]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

recover	Performs recovery and cleanup of partially-created data during an archive or restore operation. Valid with the -ds_name argument.
list	Lists the extent of recovery datasets at the current site.
update	Updates the archived published objects extinct site ID with the new site ID. Valid with the -extinct_siteID and -new_siteID arguments.

-ds_name

Specifies name of the recovery dataset. Valid with the **-f=recover** argument.

-extinct_siteID

Specifies the ID of the site previously owning the data. Valid with the **-f=update** argument.

-new_siteID

Specifies the ID of the site currently owning the data. Valid with the **-f=update** argument.

-verbose

Includes additional detailed messages in the utility's output.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in *Manually configure the Teamcenter environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- List the extent of recovery datasets at the current site:


```
ar_recover -f=list -verbose
```

- To recover from a failed *archive* operation for audit logs:

```
ar_recover -u=Tc_admin -p=password -g=group -f=recover  
-ds_name="__Recover_archive_00003f58_5e731606_Fnd0AuditArchive-19-Mar-2020  
12:19:35"
```

- To recover from a failed *restore* operation for audit logs:

```
ar_recover -u=Tc_admin -p=password -g=group -f=recover  
-ds_name="__Recover_restore_00001dc0_5e7340c5_Fnd0AuditArchive-19-Mar-2020  
15:21:22"
```

archive

Accepts input objects of type **Item** and 4GD objects of type **MdlOApplicationModel**. When the Multi-Site Collaboration archive feature is invoked, this utility prepares, transfers, and publishes the input objects and their dependents to the archive site with ownership privileges. Run this utility from the owning site. Archiving is a push operation from the owning site to the archive site. (The companion *restore* operation is a pull operation from the archive site to the owning site.)

Use caution when specifying the **-include_bom** option, as all BOM child components are archived. Consider running the utility with **-include_bom** and the **-f=list** option to evaluate the components that will be archived.

SYNTAX

```
archive -u=user-ID {-p=password | -pf=password-file} -g=group
```

```
-f={list | perform} [-item_id=itemid] [-rev=revision]
[-uidfile=data_file] [-itemidsfile=data_file] [-key=key] [-itemKeyFile=data_file]
[-include_bom] [-report_file=report_filepath] [-4gd_id] [-exclude_ids=file_name]
[-verbose] [-class=wso_class_name] [-classoffile=class_name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

- list** Performs a dry run and reports the objects that would be archived from the database.
- perform** Performs an archive and reports the objects that are archived from the database.

-item_id

Specifies the Item ID of the item to process.

-rev

The revision of the item to process.

-uidfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item UUIDs of items to process.

-itemidsfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item IDs of items to process.

-key

Specifies a comma-separated list of items or 4GD key name and value pairs. If the key is for a 4GD object, the related 4GD class must also be specified with **-class=**. Use the following format:

```
keyAttr1=keyVal1[,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

Item example:

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

4GD example:

```
4gd_id=CD000001_ID,object_name=CD000001_Name,object_desc=CD000001_Desc
```

-itemKeyFile

Specifies the name of the file containing the list of Item or 4GD object key strings to process. The file format is:

```
keyAttr1=keyVal1[,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

If the list has 4GD object key strings, the related 4GD classes must also be specified with **-classoffile=**.

File list with item IDs:

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

File list of item IDs with revisions:

```
item_id=s1005,rev_id=B
item_id=s1005-1,rev_id=A
item_id=s1005-2,rev_id=C
```

File list of 4GD objects with **-classoffile=**:

```
4gd_id=CD_Export_001
4gd_id=CD00001_ID,object_name=CD00001_Name,object_desc=CD00001_Desc
```

-include_bom

Optionally includes assembly components of the BOM to all levels for processing. This option cannot be used with 4GD objects.

-report_file

Specifies that a report be saved to the given file path. If the *report_filepath* argument is not given, the archive report is created in the folder containing the utility logs.

-4gd_id

Specifies a 4GD object identifier or 4GD object pattern to process. The **-class** or **-classoffile** argument must be specified with the **-4gd_id** argument.

The utility maps the **-4gd_id** argument to the corresponding unique ID of the 4GD class. For example:

```
Class=Cpd0CollaborativeDesign, 4gd_id=CD_100
```

A 4GD partition object uses multifold key attributes by default. Consequently, using **-4gd_id** for partition objects may result in the archiving of multiple objects.

-exclude_ids

Specifies the name of an input file listing Item ID strings of items to exclude from archiving. **-exclude_ids** cannot be used with 4GD objects.

-verbose

Includes additional detailed messages in the utility's output.

-class (cl)

Specifies the Teamcenter class of the object specified by the **-4gd_id** argument.

-classoffile (cof)

Specifies the 4GD Teamcenter class of the objects contained in the input file. Valid on with the **-itemKeyFile** argument.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in *Manually configure the Teamcenter environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility accepts input objects of type **Item** and 4GD objects of type **MdIIOApplicationModel** only.

The utility removes all replicated and published information associated with input objects and further transfers the input objects and their dependents to the archive site with ownership privileges.

You cannot archive the latest revision if an earlier revision is still present. Archiving the latest revision also archives the item.

EXAMPLES

- Generate a dry run report of an item ID and its dependent objects that would be archived:

```
archive -u=Tc-admin-user -p=password -g=group
        -f=list -verbose -item_id=ABC_101
```

Generate a dry run report for a given input file of item ID strings that would be archived:

```
archive -u=Tc-admin-user -p=password -g=group
        -f=list -itemidsfile=item_archive.txt -verbose
```

Perform an archive and generate a report for a given item ID:

```
archive -u=Tc-admin-user -p=password -g=group  
-f=perform -verbose -item_id=ABC_101  
-report_file=c:\temp\myreport.txt
```

attribute_export

Queries for the objects that satisfy the conditions specified by condition property name/property value pairs, sets new values for the properties to be updated on the found objects, and exports the data to a TC XML file. You then must use the **tcxml_import** utility to import the TC XML file containing the updated values back into the database.

Caution:

Using the bulk load feature (**-bulk_load** argument) of the **tcxml_import** utility to import a TC XML file requires the **SITCONS_AUTH_KEY** environment variable be set to a valid license key value. Due to the potential for data integrity issue if this feature is used improperly, you must meet specific criteria before you can obtain the required key from Siemens Support.

For more information about loading bulk data, see *Teamcenter Data Exchange*.

You can use these two utilities in conjunction to update the properties of numerous objects at once. The update process allows you to update one or more attributes of an object. When a specified object contains multiple attributes, only the specified attributes are exported and updated.

For complex update operations, you can create an input XML file containing instructions on which objects to update and the new values to apply. The file is constructed as a series of **UpdateSet** entries; each entry must contain **type**, **where**, and **update** components.

Before performing lengthy update operations, you can use the utility to determine the number of objects affected by a specified update operation.

For more information about the bulk update process, including instructions on creating the input file, examples of different update operations, performance statistics, and methods for managing the operation's duration, see *Teamcenter Administration*.

SYNTAX

attribute_export

```
-u=user-ID
{-p=password | -pf=password-file}
[-g=group]
{-inputfile=path-to-input-XML-file | -type=type-name}
-cond_prop=property-name
-cond_value=value-name
-update_prop=property-name
-update_value=property-value}
[-cond_operator=operator]
[performSchemaValidation]
[-queryonly]
[-log=log-file-path]
[-outdir=output-XML-file-directory]
```

`[-batchsize=number-of-objects-to-update-per-batch]`
`[-islands=number-of-objects-to-update-per-island]`
`[-untransformed]`
`[-uidfile]`
`[-h]`

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. The group value must be **dba** to run this utility.

-inputfile

Specifies the full path to the input XML file containing instructions on which objects to update and the new values to apply. The file is constructed as a series of **UpdateSet** entries; each entry must contain **type**, **where**, and **update** components.

For more information about creating the input file, see *Teamcenter Administration*.

Use the input file for complex updates. (For simpler instances, you can use the following arguments.)

You must use either this argument, or the **-type**, **-cond_prop**, **-cond_value**, **-update_prop** and **-update_value** arguments.

-type

Specifies the type of objects to be updated. For example, **item** or **dataset** or **StructureContext**, and so on.

This argument accepts a single value, which must be a valid Teamcenter object. Use multiple instances of this argument to specify multiple object types.

You must either use this argument in conjunction with the **-cond_prop**, **-cond_value**, **-update_prop** and **-update_value** arguments, or use the **-inputfile** argument.

-cond_prop

Specifies the name of the property to be queried for and updated during export.

This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object. For example:

-cond_prop=object_name, last_mod_date

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-cond_value** argument.

You must either use this argument in conjunction with the **-type**, **-cond_value**, **-update_prop** and **-update_value** arguments, or use the **-inputfile** argument.

-cond_value

Specifies the new value for the property specified by the **-cond_prop** argument.

This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object. For example:

-cond_value=TextData_es_ES, "01-DEC-2011 00:00"

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-cond_prop** argument.

You must either use this argument in conjunction with the **-type**, **-cond_prop**, **-update_prop** and **-update_value** arguments, or use the **-inputfile** argument.

-update_prop

Specifies the internal name of the property to be updated (as opposed to the display name). For example **object_desc**, **char VLA**, and so on.

This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object. For example:

-update_prop=object_name, object_desc

You can use multiple instances of this argument. Each instance of this argument should be paired with the **-update_value** argument.

You must either use this argument in conjunction with the **-type**, **-cond_prop**, **-cond_value**, and **-update_value** arguments, or use the **-inputfile** argument.

-update_value

Specifies the new value for the property specified by the **-update_prop** argument.

This argument accepts multiple values in a comma-separate list. For example:

-update_value=folder, "Home folder"

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-update_prod** argument.

You must either use this argument in conjunction with the **-type**, **-cond_prop**, **-cond_value**, and **-update_prop** arguments, or use the **-inputfile** argument.

-cond_operator

Specifies the operator to be used with the condition arguments. Valid operations are: **equal**, **not equal**, **greater than**, **greater than and equal**, **less than**, **less than and equal**.

When setting the value for this argument in the command window, the format for these operations are: **EQ**, **NE**, **GT**, **GE**, **LT**, and **LE**.

Note:

When setting the value for this argument in the input file, you can use the above format, or the following characters: **=**, **!=**, **>**, **>=**, **<**, **<=**.

This argument is optional and is only valid when used with, and placed after, the **-cond_prop** and **-cond_value** arguments. For example:

```
-cond_prop=object_name -cond_value="My obj #1"
-cond_prop="last_mod_date" -cond_value="01-DEC-2011 00:00"
-cond_operator="LE" -cond_prop="last_mod_date"
-cond_value="01-DEC-2010 00:00" -cond_operator="GE"
```

-performSchemaValidation

Enables schema validation of the XML input file.

-queryonly

Outputs the number of target objects affected by the specified update parameters to a log file. If the number of objects is less than 100, the object UIDs are included in the log file.

When you specify this switch, the utility does not perform the update, it merely reports the number of affected objects.

Use this switch in conjunction with either the input file or the condition and update arguments to determine how many objects are affected by specified update operation. You can use the resulting information to determine batch size, and to estimate the duration of the update operation.

For more information about update duration, see *Teamcenter Administration*.

-log

Specifies the path to the log file. By default, this is the current directory.

-outdir

Specifies the directory to which the TC XML file is generated. The file name is automatically generated, using the following naming convention:

BulkAttrOut_1.xml, BulkAttrOut_2.xml, BulkAttrOut_3.xml

This argument is optional. If not specified, the current directory is used.

-batchsize

Specifies the number of objects to update per TC XML file.

This argument is optional. The default value is **800**.

-islandsize

Specifies the number of objects to update per island. An island ties logically related objects together.

For example, a simple island is:

[Item, ItemRev, MasterForm]

The data in low level TC XML is grouped into islands by closure rules, in which the root objects are assigned island IDs and the traversed objects are assigned new island IDs when an **INTER_ISLAND** clause is evaluated.

The following example illustrates an assembly in which every child component is placed in a separate island. The **I** at the end is the predicate for the **INTER_ISLAND** clause.

```
[TYPE.PSOccurrence:CLASS.WorkspaceObject:ATTRIBUTE.child_item:
PROCESS+TRAVERSE:$opt_entire_bom=="&quot;true&quot; &amp;&amp;
$opt_hl_tie=="&quot;false&quot;;I]
```

An example of how island IDs are written in low level TC XML is:

```
<ItemRevision elemId="id7" island_id="4" item_revision_id="A"
items_tag="Q9L5sgKI4ghuD" last_mod_date="2011-11-15T07:57:46Z"
object_desc="new desc" object_name="my data"
object_type="ItemRevision" owning_group="#id4"
owning_user="#id2" parent_uid="Q9L5sgKI4ghuD"
puid="gBB5sgKI4ghuD" />
```

```
<Item elemId="id7" island_id="4" puid="asd876jerTf"
last_mod_date="2011-11-15T16:03:30Z" object_desc="new description"
item_id="003293"/>
```

```
<Item elemId="id8" island_id="4" puid="xgr23hgfg43"
last_mod_date="2011-11-15T16:03:30Z" author="John Yates"
item_id="003423"/>
```

This argument is optional. The default value is **100**.

-untransformed

Exports the original values of the specified attributes.

This argument is optional.

-uidfile

Specifies the full path to the input XML file containing a list of object UIDs.

This argument is optional. If specified, it suppresses the function of the **-cond_prop** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

The IDSM server must be running when you use this utility.

You must use either the **-inputfile** argument, or the **-type**, **-update_prop**, **-update_value**, **-cond_prop** and **-cond_value** arguments.

Note:

The condition arguments are the equivalent of the where clause in an SQL phrase. Running this utility without specifying the **-cond_prop** and **-cond_value** arguments it technically possible, but impractical.

EXAMPLES

- To update the attributes of certain properties as specified in the **propFile.xml** file, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group -inputfile=
d:\propFile.xml -log=logFile.txt
```

- To update **prop3** to **value3** and **prop4** to **value4** for **type1** objects, when **prop1** equals **value1** and **prop2** is greater than **value2**, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group
-type=type1 -cond_prop=prop1 -cond_value="value1" -cond_prop=prop2
-cond_value="value2" -cond_operator="GT" -update_prop=prop3
-update_value="value3" -update_prop=prop4
-update_value="value4"
```

- To update **prop3** to **value3** and **prop4** to **value4** for **type1** objects, when **prop1** equals **value1** and **prop2** is greater than **value2** in batches of 400, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group -type=type1
-cond_prop=prop1 -cond_value="value1"
-cond_prop=prop2 -cond_value="value2" -cond_operator="GT"
-update_prop=prop3 -update_value="value3" -update_prop=prop4
-update_value="value4" -batchsize=400
```

- To update an **object description** to **red Desc** and the **object name** to **This is a new object name** for all items where the **object_name** property is **my object** and the **str_attr** property is **red,white,yellow**, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group
-type=Item -update_prop=object_desc -update_value="red Desc"
-update_prop=object_name -update_value="This is a new obj name"
-cond_prop=object_name -cond_value="my obj" -cond_value="str_attr"
-cond_value="red,white,yellow"
```

- To update an **object description** to **blue Desc** and the **int_VLA** to **10,3,0,99** for all datasets where the **object_name** property is **TextData_es_ES** and the **last_mod_date** property is **01-DEC-2011 00:00** or greater, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group
-type=Dataset -update_prop=object_desc -update_value="blue Desc"
-update_prop="int_VLA" -update_value="10,3,0,99"
-cond_prop=object_name -cond_value="TextData_es_ES"
-cond_prop="last_mod_date" -cond_value=
"01-DEC-2011 00:00" -cond_operator="GE"
```

- To update the **char_VLA** to **w,y,d,k,a** for all items where the **owning_user** property is **AsL5bfuW4ghudD**, enter the following command on a single line.

```
attribute_export -u=Tc-admin-user -p=password -g=group -type=Item  
-update_prop="char VLA" -update_value="w,y,d,k,a"  
-cond_prop=owning_user -cond_value="AsL5bfuW4ghudD"
```

appmodel_fix_scope

Fixes the scope information for imported 4th Generation Design (4GD) objects at the target site. You must run this utility the target site after you import a low-level TC XML file that contains 4GD objects. For site consolidation activities, it should be run after each import that contains 4GD objects and it must be run before any objects have their ownership transferred.

SYNTAX

```
appmodel_fix_scope -u=user-ID {-p=password | -pf=password-file} [-g=group]
[-fix_scope_for_replicas]
[-log][-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If this argument is used without a value, the operating system user name is used.

Note:

If your Teamcenter server uses Security Services single sign-on, see [Before you begin](#) for additional information.

-p

Specifies the user's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-fix_scope_for_replicas

Fixes the scope information for imported replica model elements and artifacts imported in a low-level TC XML file.

-log

Specifies the name of the file containing the log entries for the utilities actions. If you do not supply this argument, the utility saves the log entries in the **fixScopeLog.log** file.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility does not function if the ownership of 4GD replica objects has been transferred to the target site. It must be run before any ownership transfers take place.

EXAMPLES

- To fix the scope information for replica model elements and artifacts and log the actions in the **Site2Import.log** file:

```
appmodel_fix_scope -u=Tc-admin-user -p=password -g=group
-fix_scope_for_replicas -log=Site2Import.log
```


batch_export_translate_import

Exports, translates, and/or imports the named reference of a given type attached to the specified dataset type that is attached to a specified item revision with the given relation type. For example, this utility could be used to export **.wire** files (named references) from the **ALIAS_PROJECT** dataset attached to a specified **CORP_CriteriaRevision** item revision with an **IMAN_specification** relation to the directory specified by the **-output_path** directory.

This utility works in one of the following modes:

- Export mode (**-e**)

Exports datasets from Teamcenter.

- Import mode (**-i**)

Imports datasets to Teamcenter.

- Export, translate, import mode (**-eti**)

Exports, translates, and imports the dataset back in to Teamcenter.

SYNTAX

```
batch_export_translate_import [-u=user-id {-p=password | -pf=password-file}
-g=group] -infile=input-file -output_path=output-path
-translator=translator-executable -e -i -eit -nolog [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-infile

Specifies the input file for exporting, translating, and importing a list of datasets.

-i

Specifies that the utility performs only the batch import. The input file format is as follows:

```
item-id | rev-id | dataset-type | relation | reference-type | dataset-uid |
new-dataset-name | path-of-file-to-be-imported
```

-e

Specifies that the utility performs only batch export. The file format for exporting a given list of datasets is as follows:

```
item-id | rev-id | dataset-type | relation | reference-type | dataset-uid
```

-eti

Specifies that the utility export, translate, and import a list of datasets. The file format for exporting a given list of datasets is as follows:

```
item-id | item_rev-id | dataset-type | relation-type | reference-name | dataset-uid
```

-translator

Specifies the translator executable or batch file used to translate the exported files.

-nolog

Specifies that a log file will not be generated when the utility is run.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*. In addition, the log file is created in the local directory specified by either the **TMP** or **TEMP** environment variable.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To export the list of given datasets to the **C:\temp** directory, enter the following command on a single line:

```
batch_export_translate_import -u=user-name -p=password -g=group-name -e
-infile=input-file-with-dataset-details -output_path=c:\temp
```

- To import the list of given datasets, enter the following command on a single line:

```
batch_export_translate_import -u=user-name -p=password -g=group-name -i
-infile=input-file-with-dataset-details -output_path=c:\temp
```

- To translate the list of given datasets, enter the following command on a single line:

```
batch_export_translate_import -u=user -p=password -g=group-name
-eti -infile=input-file-with-dataset-details
-translator=input-translator-file -output_path=c:\temp
```

briefcase_validator

Briefcase files must be created using a Siemens Digital Industries Software application. Data in Briefcases created using unsupported third-party applications (such as with an archiving application and then renamed as a .bcz file) cannot be imported. **briefcase_validator** lets you quickly validate that a Briefcase file was created using a Siemens Digital Industries Software application and not a third-party application. Validating the integrity of a Briefcase file before attempting to import the data minimizes import errors and reduces the likelihood of needing to re-import data.

By default, **briefcase_validator** lists its results in the command window. You can optionally send the results to an HTML file for viewing.

For more information about validating Briefcase TC XML files, see [tcxml_export](#) and Transferring a briefcase package.

You can also use the **briefcase_validator -file** argument to create Briefcase files based on existing Briefcase files, but containing changed data and metadata. Doing so can be useful when sharing data with other organizations for testing purposes. For example, the receiving organization can update the text-based XML data in the Briefcase file to match their data model and site names. They can then import and review the Briefcase data.

SYNTAX

```
briefcase_validator -u=user-ID {-p=password | -pf=password-file} -g=group
[-inputfile=path-to-input-XML-file
[-ruleFile=rule_file_name] [-show] [-report=report_name.html]]
[-file=file_name
{-add=file_to_add | -dir=dir_name}
[-props=srcsite_id?destsite_id?tcxml_file_name[?briefcase_type]]
[-ll_bcz_mode]]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the user's password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument.

For more information about managing password files, see [Manage password files](#).

-g

Specifies the group associated with the user. The group value must be **dba** to run this utility.

-inputfile

Specifies the full path to the Briefcase or TC XML file to be validated.

-ruleFile

The mapping rule file to use when validating the file specified by **-inputfile**.

-show

Displays the contents of the file specified by **-inputfile**.

-report

Specifies a file to be created containing the validation report in the current directory if a full path is not specified. Use the extension *.html* on the file name. **-report** is used with **-inputfile**.

-file

Specifies the full path and file name of the Briefcase file to be created.

-add

Specifies the attachments to package in the Briefcase file being created. Separate multiple attachments with commas. Include the full absolute path and file name of each attachment.

-add is used with **-file** and cannot be used with **-dir**.

-dir

Specifies a directory containing attachments to package in the Briefcase file being created. Include the full absolute path of the directory.

-dir is used with **-file** and cannot be used with **-add**.

-props

Specifies property values to set in the new Briefcase manifest file. Property values are separated by ? characters. Used with **-file**.

Property values must be set in the following order.

- *srcsite_id* specifies the site ID of the source Teamcenter site. This value must match the source site ID specified in *tcxml_file_name*.
- *destsite_id* specifies the site ID of the target Teamcenter site. This value must match the target site ID specified in *tcxml_file_name*.
- *tcxml_file_name* specifies the Briefcase XML file containing the Teamcenter data to include in the new Briefcase file.
- *briefcase_type* specifies the type of Briefcase file being created. Values are case-sensitive.

Value	File created
M2M (the default)	Managed Briefcase file
M2U	Unmanaged Briefcase file generated at a Teamcenter site and targeted for a Briefcase Browser site
U2M	Unmanaged Briefcase file generated at a Briefcase Browser site and targeted for a Teamcenter site
U2None	Unmanaged Briefcase file

-ll_bcz_mode

Specifies that the Briefcase file be created as a low-level Briefcase file. Used with **-file**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

This utility validates the low-level TC XML in an existing Briefcase file or other low-level TC XML file. High-level TC XML is not supported.

EXAMPLES

- List the contents of the specified Briefcase file.

```
briefcase_validator -u=Tc-admin-user -p=password -g=group
-file= c:\temp\sample_briefcase.bcz -show
```

- Evaluate the exported TC XML in an existing Briefcase file.

```
briefcase_validator -u=Tc-admin-user -p=password -g=group
-inputfile= c:\temp\sample_briefcase.bcz
-report=c:\temp\validation_report.html
```

- Evaluate a stand-alone low-level TC XML file.

```
briefcase_validator -u=Tc-admin-user -p=password -g=group  
-inputfile=c:\temp\sample_tcxml.xml  
-report=c:\temp\validation_report.html
```

- Create a new Briefcase file using data changed from a source Briefcase file.

```
briefcase_validator -u=Tc-admin-user -p=password -g=group  
-file= c:\temp\new_briefcase.bcz  
-dir=c:\temp\new_bc001\  
-props=vender01?oemtest01?trans6216b0f0000420?U2M
```

briefcase_ownership_recovery

Recovers ownership of objects transferred to another site associated with a briefcase or when transferred in other transactions. Object ownership is returned to the original (exporting) site and objects on the remote (importing) site are reverted to being replicas. Ownership can be recovered in cases where ownership was previously transferred using a release of Teamcenter that includes the **briefcase_ownership_recovery** utility.

Caution:

To ensure data integrity, run this utility at both the exporting site and the importing site.

The utility generates an output file listing details of all objects for which ownership is recovered successfully.

SYNTAX

```
briefcase_ownership_recovery -u=user-ID {-p=password | -pf=password-file} -g=group
[-briefcase=file | -transactionid=transaction_id | -siteid=site_id]
[-mode= import | export]
[-startdate=start_date]
[-enddate=end_date]
[-report]
[-outputDir=dir_path]
[-delete]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. If this argument is used without a value, the operating system user name is used.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-briefcase

Specifies the briefcase or TC XML file for which you want to recover ownership. Ownership of all associated objects with transferred ownership are recovered. Specify one briefcase or TC XML.

Do not use **-briefcase** if either **-transactionid** or **-siteid** is specified. If **-transactionid**, **-briefcase**, or **-siteid** are not specified, **-briefcase** is used.

-transactionid

Specifies the ID of the transaction for which you want to recover ownership. Ownership of all associated objects with transferred ownership will be recovered. Specify one transaction id.

Do not use **-transactionid** if either **-briefcase** or **-siteid** is specified.

Locate the transaction ID in the transaction export log or in the header section of the TC XML in the briefcase.

-siteid

Use **-siteid** to review and select a briefcase from all briefcases exported or imported for a particular site. **-siteid** specifies the ID of the site for which all briefcases exported and imported by the current user are listed. If the current user is a member of the dba group, briefcases exported or imported by all users are listed. Results are listed by date

Each briefcase in the list is preceded by an index number. When prompted, enter the index number of a briefcase to recover the ownerships transferred with that briefcase.

Use **-siteid** with **-mode**, **-startdate**, and **-enddate** to filter the list of briefcases.

Do not use **-siteid** if either **-briefcase** or **-transactionid** is specified.

-mode

Use with **-siteid** to limit the list of briefcases to only those exported or imported. Allowed values are **export** or **import**.

-startdate

Specifies the earliest date and time a transaction must have occurred to be included in the list generated with the **-siteid** option.

The format of the date is “DD-MMM-YYYY HH:MM:SS”. Double quote marks are required. For example, "21-Jun-2019 22:15:00".

-enddate

Specifies the latest date and time a transaction must have occurred to be included in the list generated with the **-siteid** option.

The format of the date is “DD-MMM-YYYY HH:MM:SS”. Double quote marks are required. For example, "22-Jun-2019 22:15:00".

-report

Displays a list of the objects for which ownership will change if you run the current command. For each object, the list includes its unique ID, type, owning site ID, and whether it is a principle object.

When **-report** is specified, ownership changes do not occur.

-outputDir

Specifies the file path where the report file is stored. If **-outputDir** is not specified, the report file is stored in the directory where you ran the utility.

-delete

Deletes the ownership recovery record from the database for the specified transaction. Once deleted, **briefcase_ownership_recovery** cannot recover ownership for the specified transaction.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in *Manually configure the Teamcenter environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

briefcase_ownership_recovery recovers ownership transfers only for briefcases created using a release of Teamcenter that includes the **briefcase_ownership_recovery** utility.

EXAMPLES

- Recover ownership of objects in a specific briefcase.

```
briefcase_ownership_recovery -u=username -p=password -g=dba
-briefcase=BCZ_test1__IMC-10001_27-Jun-2019-12_10_42.bcz
```

- Report the objects for which ownership would be recovered in a briefcase without actually recovering ownership of the objects.

```
briefcase_ownership_recovery -u=username -p=password -g=dba
-briefcase=BCZ_test1__IMC-10001_27-Jun-2019-12_10_42.bcz -report
```

- Recover ownership of objects in a TC XML file on a site that exported the briefcase.

```
briefcase_ownership_recovery -u=username -p=password -g=dba
-briefcase=OffExp5bfce85700004c3000007bf5.xml
```

- Recover ownership of objects from a specific transaction.

```
briefcase_ownership_recovery -u=username -p=password -g=dba
-transactionid=Prod34-1000158101227Nov2018
```

- List exported briefcases from a particular month and recover ownerships from one of the listed briefcases.

```
briefcase_ownership_recovery -u=username -p=password -g=dba
-siteid=12345 -mode=export -startdate=01-jun-2019 -enddate=30-jun-2019
```

Following is an example list:

```
1. Model102_IMC-759_27-Jun-2019-12_10_42.bcz | jjohnson01 | 1030 | 27-Jun-2019
2. Model099_IMC-dkc_21-Jun-2019-08_43_15.bcz | jjohnson01 | 1030 | 21-Jun-2019
3. Model087_IMC-001_21-Jun-2019-08_35_32.bcz | jjohnson01 | 1030 | 21-Jun-2019
```

Entering **2** at the command prompt recovers ownership of the objects transferred to another site associated with briefcase Model099_IMC-dkc_21-Jun-2019-08_43_15.bcz.

cleanup_ic_objects

Reverts changes made to an assembly as part of an incremental change (IC) object. This utility analyzes the incremental change object and cleans up the database to nullify the impact of the IC changes. For example, if a PLM XML file that is updating an existing structure in the database is imported using an IC object, this utility can be used to revert the effect the import. This can be very helpful in cases where the PLM XML import failed and the data is not in a consistent state.

Note:

This utility does not restore the database to its original state before the import.

This utility can revert the following changes:

- Adding, removing, or modifying a BOM line.
 - Add an existing item as a BOM line.
 - Add an existing item as a BOM line and occurrence group child.
 - Add an existing item with GDE children as a BOM line and occurrence group child.
 - Add an existing connection as a BOM line and occurrence group child and connect it to GDE BOM lines.
 - Modify a BOM line property.
 - Remove an item BOM line.
 - Remove an item BOM line with GDE children.
 - Remove a connection BOM line.
- Adding, removing, or modifying a GDE line.
 - Add a GDE line.
 - Modify a GDE line property.
 - Remove a GDE line.
- Adding, removing, or modifying attachments (forms/datasets).
 - Attach a form/dataset to an item BOM line.

- Attach a form/dataset to an item BOM line in absolute occurrence context.
- Attach a form to a GDE line in absolute occurrence context.
- Attach a form to a connection BOM line in absolute occurrence context.
- Modify a BOM line form/dataset.
- Modify a BOM line form/dataset in absolute occurrence context.
- Remove a BOM line form/dataset.
- Remove a BOM line form/dataset in absolute occurrence context.

A given IC object may contain many ICEs (incremental change elements). Each ICE specifies the change type that was made (add/remove) and the affected object of the change.

For add scenarios, the change type is **IC_add**.

- When adding a BOM line/GDE line, the affected object is **PSOccurrence/GDEOccurrence**. This utility deletes the ICE and **PSOccurrence/GDEOccurrence**.
- When attaching a dataset/form to a BOM line, the affected object is **ImanRelation**. This utility deletes the ICE and **ImanRelation**. The secondary objects of the relation are not deleted.

For modify scenarios, the change type is **IC_add**.

- When modifying a BOM line property, the affected object is **AbsOccData**. This utility deletes the **AbsOcc** data.
- When modifying a BOM line attachment, the affected object is a copy of the attachment (dataset/form) with the modifications. This utility deletes the modified copy.

For remove scenarios, the change type is **IC_remove**.

- The affected objects point to the objects that are removed from the structure. This utility deletes the ICE. It automatically restores the removed objects.

SYNTAX

```
cleanup_ic_objects [-u=user-id {-p=password | -pf=password-file} -g=group]
{ [-ic_id=IC-object-item-ID | -key=IC-object-search-key] [-ic_rev_id=IC-object-rev-ID] } [-dryrun] [-
verbose] [-start_date=start-date] [-end_date=end-date]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ic_id

Specifies the item ID of the incremental change object whose associated data will be deleted.

-key

Specifies the search key of the incremental change object. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-ic_rev_id

Specifies the revision ID of the incremental change object whose associated data will be deleted.

If this argument is not specified, the latest revision will be used.

-dryrun

Deletes nothing. Objects are searched and a report is created.

-verbose

Prints extra information about the objects being deleted.

You can combine the **-dryrun** and **-verbose** options as follows:

- No option: Prints errors only.
- **-verbose** option only: Prints maximum information, such as what is being searched, how many objects were found, type of objects being deleted, and success/failure message after deletion.
- **-dryrun** option only: Prints the number of objects found and the type of objects that will be deleted in a normal run.
- Both options: Prints all of the information from the **-verbose** option except the success/failure message.

-start_date

Changes made after the specified date and time are reverted. Use the format **DD-MMM-YYYY HH:MM:SS**. For example, **01-Jan-1970 00:00:00** is January 1, 1970.

If no time is specified, 00:00:00 is used.

-end_date

Changes made after the specified date and time are reverted. Use the format **DD-MMM-YYYY HH:MM:SS**. For example, **01-Jan-1970 00:00:00** is January 1, 1970.

If no time is specified, 00:00:00 is used.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

convert_replica_files_to_stubs

Converts existing replica Teamcenter file (**ImanFile** objects) to **pom_stub** objects at the specified remote site, all remote sites, or specified objects at all remote sites; it purges the corresponding operating system volume files to conserve space. It can also populate the file server cache (FSC) with replica files.

SYNTAX

```
convert_replica_files_to_stubs [-u=admin-id {-p=password | -pf=password-file} -g=dba]
{[-by_site_name=remote-site-id | ALL] | [-by_plmxml=file-name]} [-verbose]
[-populate_cache] [-query ] [-batch_size=size-value]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. The group value must be **dba** to run this utility.

-by_site_name

Specifies the name of the remote site where replica files are replaced with stubs. If **ALL** is specified, replica files at all remote sites are replaced. This argument cannot be combined with the **-by_plmxml** argument.

-by_plmxml

Specifies the name of a PLM XML file that contains replica objects to be replaced by stubs. Only the objects listed in the file are replaced.

You must generate the PLM XML file specified in this argument using the **ConfiguredDataExportDefault** or **justDatasetsOut** transfer modes. Other transfer modes are not supported.

This argument cannot be combined with the **-by_sitename** argument.

-verbose

Includes additional information about the process in the utility's output.

-populate_cache

Populates the FSC with replica files at the sites where replica objects are replaced with stub objects.

-query

Returns the number of objects that will be converted to stubs.

-batch_size

Specifies the number of objects in each batch that is processed. If this argument is omitted, all objects are processed in a single batch.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

The IDSM server must be running when you use this utility.

EXAMPLES

- Replace all replica objects at all sites with stub objects:

```
convert_replica_files_to_stubs -u=Tc-admin-user -p=password -g=group
-by_site_name=ALL
```

- Replace replica objects at the cologne site and populate the site's FSC:

```
convert_replica_files_to_stubs -u=Tc-admin-user -p=password -g=group
-by_site_name=cologne -populate
```

- Determine the number of replica objects that will be replaced at a site:

```
convert_replica_files_to_stubs -u=Tc-admin-user -p=password -g=group
-by_site_name=cologne -query
```

- Replace all replica objects at all sites with stub objects processed in batches of 200 objects:

```
convert_replica_files_to_stubs -u=Tc-admin-user -p=password -g=group  
-by_site_name=ALL -batch_size=200
```

- Replace replica objects in the **stub_replicas.xml** file at all sites with stub objects processed in batches of 200 objects:

```
convert_replica_files_to_stubs -u=Tc-admin-user -p=password -g=group  
-by_plmxml=stub_replicas.xml -batch_size=200
```

database_verify

Compares database schema, Teamcenter types, tools, release statuses, and units of measure between two specified Multi-Site Collaboration sites and generates a report of any database discrepancies.

You can use this utility to query types and classes from a specified remote site and create a local dataset named **TCTYPES_SITE***siteid* containing those mappings. There is one dataset for each remote site defined with the **-site** argument. You can also create and update the local dataset of remote class and types mappings for a specified remote site. Run this utility whenever there are changes for the POM transmit file of a specified site.

SYNTAX

```
database_verify [-u=user-id {-p=password | -pf=password-file} -g=group]
               -from=site-name1 -to=site-name2
               [-schema] [-type] [-tool] [-status] [-uom] [-all] [-output=file-name]
               [-site=site-name] -force -offline [-filename=file-name] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-from

Specifies a Teamcenter database site to be verified.

-to

Specifies a Teamcenter database site to be verified.

-output

Specifies the output format. The report is output to a file if a file name is specified. If not, the report is displayed in a shell.

-schema

Compares schema between the two sites.

-type

Compares types between the two sites.

-tool

Compares tools between the two sites.

-status

Compares release status types between the two sites.

-uom

Compares units of measure between the two sites.

-notetype

Compares note types.

-all

Compares classes, types, tools, status types and units of measures between the two sites. This is the default if no argument is supplied.

-site

Specifies the site name where types and classes would be persisted locally. If the value of this argument is set to **ALL**, the utility generates these datasets for all sites in the database.

-force

Generates the type-class mappings file even when the POM transmit files for the remote and local sites have not changed.

-offline

Specifies the site identified with the **-site** argument is offline. If you specify this argument, you must also specify the **-filename** argument.

-filename

Specifies the file name generated by this utility from the **-site** argument. This argument is required if the **-offline** argument is specified.

-v

Runs utility in verbose mode. Displays maximum amount of information. Typically, nonverbose utility sessions only display error messages.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

- To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
- The **-from** and **-to** arguments must be specified.

data_share

Used for various Multi-Site Collaboration operations, such as publishing and unpublishing objects collectively and sending objects to remote sites. It can be used as a deployment tool during the initial Multi-Site Collaboration implementation phase or as a day-to-day tool for performing functions that previously were available only through the user interface. This utility is especially helpful in setting up and maintaining a hub configuration.

The behavior of this utility is controlled by the **TC_force_remote_sites_exclude_files** preference. If this preference is set to **true**, the replica files are stored in the remote site FMS server cache (FSC); otherwise, the replica files are stored in the remote system volume.

The behavior of the utility for project relationships on replica objects can be controlled by the **TC_sync_projects_with_owning_site** preference. This preference is not created by the Teamcenter installation process. To change the default behavior of shared project relationships, you must create the preference.

The utility also supports TC XML transfers for of 4th Generation Design (4GD) data. The 4GD relation data mapping is controlled by the **TC_cms_relation_optset_map** preference. Use this preference when you want to control the relations that are included or excluded when replicating a 4GD object.

This utility supports part family templates and part family members. Use this utility to:

- Mass publish objects to one or more Object Directory Services (ODS) sites.
- Mass unpublish objects from one or more ODS sites.
- Publish or unpublish an entire assembly.
- List ODS sites currently defined in the local database and authorized for publication.
- Send objects to other sites.
- Delete obsolete publication records at the ODS.
- Check current status of authorized publication sites.
- List ODS sites to which an object is published.
- Import an item from a remote site.
- Export 4GD data in TC XML format.

Data can be input to this utility in the following forms:

- Input file

- Folder name
- Object ID template

When sending objects to a specific user and/or group at a remote site using the **-owning_user** and **-owning_group** arguments, the following rules apply:

- If both the specified user and group exist at the importing site, the imported objects are owned by the user and group regardless of whether or not the user is a member of the group.
- If only the user is specified or if the group is specified but does not exist at the importing site, the user's default group at the importing site is the owning group of the imported objects.
- If only the group is specified or if the user is specified but does not exist at the importing site, the user context of the remote IDS process is the owning user of the imported objects.

AVOIDING OUT-OF-MEMORY ERRORS

To avoid possible out-of-memory errors when you are replicating architecture revisions, Siemens Digital Industries Software recommends you exclude **MEAppearancePathNode** (APN) objects by setting the following environment variable:

```
TC_EXCLUDE_APN=TRUE
```

This excludes APN objects from the export or import. It may also exclude associated object, such as JT for promoted and deformed bodies. You can replicate the APN objects using the **sync_product_apns** utility.

If you do not set this environment variable, or you set its value to **FALSE**, you can use the batch feature of this utility to overcome memory limitations when replicating large numbers of APNs. For example, you may specify:

```
-batch_objects=MEAppearancePathNode -batch_size=5000
```

When setting the batch size, consider the number of APNs generated by your business processes and the amount of memory available on your system.

SYNTAX

```
data_share [-u=user-id {-p=password | -pf=password-file} -g=group]
-f={function} [-site=remote-site-name1 -site=remote-site-name2... ]
[-owning_user=remote-user] [-owning_group=remote-group]
{-item_id={item-id | template} | -folder=folder-name |
-name=workspace-object | -filename=input-file |
[-key=keyAttr1=keyVal1,keyAttr2=keyVal2...,keyAttrN=keyValN |
-itemKeyFile=file-name] | [-itemRevisionKeysFile=file-name]]
[-class=wso-class-name | -classoffile=class-name]
[-include=relation-type1 -include=relation-type2...]
```



```

[-exclude=relation-type1 -exclude=relation-type2...]
[-revision-selector | -rev=rev-id ]
[-include_bom] [-include_modified_only]
[-assert_precise] [-transfer] [-attach]
[-exclude_files] [-latest_ds_version] [-exclude_folder_contents]
[-include_bc] [-include_supercedures] [-include_pfmembers]
[-include_pftemplates] [-pf_bom_treatment=option]
[-qry_name=query-name -qry_attr=attr-name1 -qry_val=attr-value1[-qry_attr=attr-name2
-qry_val=attr-value2...]]
[-oat] [-continue_on_error] [batch_size=number-objects-per-batch]
[-report=report-file-name] [-user=user-id] [-group=group-name]
[-error_file=error-file-name] [-exclude_variant_options] [-dir]
[-mapping_rule=rule-file-name]
[-batch_variant_options] [-batch_objects=class-for-deferred-objects]
[-batch_file=file-name-for-deferred-objects]
[-include_dist_comp] [-lightweightobject] [-log]
[-checkpoint [-compress_ind_files=S | I | N ] [-transaction_id=transaction-id]
[-tcx_data_load [-migration_sync]]
[-status]
[-cleanup_transaction [-transaction_id=transaction-id | -before_last_process_date=date] ]
[-restart] [-commit_ixr] [-list_transactions]
[-optionset | -optionset=optionset-name [-de_incl_rlz_bom]
[-workset_include_relz_de] [-4gd_id=object-id -class=4gd-class-name] ]
[-all_roles] [-all_subgroups] [-all_groupmembers]
[-include_4gd_baseline_content]
[-parallelize=owner-processes,remote-processes]
[-override_options=option1:value1,option2:value2, ...]
[-session_options=option1:value1,option2:value2, ...]
[-volume_file_list=volume_list]
[-h]

```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID. This is generally a user with administration privileges. Be aware that when **data_share** is run by a system administrator, **Has Bypass ACL** is set and access control list rules are bypassed.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

Caution:

For HTTP enabled sites, remote site operations log on using the default group for the user supplied with the **-u** argument. Any value supplied with the **-g** argument is ignored.

-f

Specifies the function to be performed; you must specify one of the following:

send

Sends objects to the specified remote sites.

The objects to send are determined by the **-item_id**, **-folder**, **-filename**, **-4gd_id**, **-all_subgroups**, **-all_roles**, or **-all_groupmembers** arguments.

publish (pub)

Publishes objects to the given ODS sites. The objects to publish are determined by the **-item_id**, **-folder**, or **-filename** arguments. Cannot be used with the **-all_subgroups**, **-all_roles**, or **-all_groupmembers** arguments.

unpublish (unp)

Unpublishes objects from the given ODS sites. The objects to unpublish are determined by the **-item_id**, **-folder** or **-filename** arguments. Cannot be used with the **-all_subgroups**, **-all_roles**, or **-all_groupmembers** arguments.

delete_pubrec (dpr)

Deletes obsolete publication records for the specified object from the local database. This must be run at the ODS site containing the publication record to be deleted. Only privileged users may use this function. Requires the **-item_id** argument with specific item ID; no wildcards or other arguments are supported with this function.

Note:

To be used only if the primary object has been deleted but publication records still exist at the ODS site.

register (reg)

Registers item IDs to the central item ID registry.

unregister (unreg)

Unregisters item IDs from the central item ID registry. The register and unregister functions must be supplied with the **-item_id** or **-filename** argument. To unregister deleted items, you must run this utility at the central item ID registry site.

delete_exprec (dxr)

Deletes export records for the specified sites for objects listed in the text file identified by the **-filename** and **-classoffile** arguments. It does not traverse item structure. Only privileged users may use this function.

Note:

To be used only as a last resort after attempting to delete export records using the **-verify** argument of the **data_sync** utility.

list_ods (lo)

Lists the authorized ODS sites, which consist of the default ODS site and the sites specified by the **ODS_publication_sites** preference.

check_ods (co)

Lists the availability of authorized ODS sites.

list_pub_info (lpi)

Lists publication information about objects. Must be run at the owning site.

find_duplicates (fd)

Compares all of the item IDs at the remote site specified by the **-site** argument. The item IDs searched for may be filtered with the **-item_id**, **-created_before**, and **-created_after** arguments. The output may be directed to a file using the **-report** argument. The output is formatted to **csv** style, using comma-separated values.

-created_before

Restricts searches for duplicate items to those created at the target site before the specified date.

-created_after

Restricts searches for duplicate items to those created at the target site after a specified date.

remote_checkout

Check out replica objects on a remote site, for example,

```
data_share -f=remote_checkout -filename=uid.txt -
classoffile=Tagstring
```

checkin_replica_co

Check in replica objects on a remote site. Specify the object to check in with **-rep_co_tag**. Must be used with **-session_options** as shown in the following example.

```
data_share -f=checkin_replica_co -rep_co_tag=obj_uid -
session_options=debug:true
```

list_remote_co (lremco)

Lists primary objects that are checked out by remote users based on the specified user ID, group name, and site name.

If no user, group, or site is specified, all remote checkouts are listed.

list_replica_co (lrepco)

Lists replica objects that are checked out from a remote site based on the specified user ID, group name, and site name.

If no user, group, or site is specified, all replica checkouts are listed.

cancel_remote_co (cremco)

Cancels all remote checkouts based on the specified user ID, group name, and site name.

If no user, group, or site is specified, all remote checkouts are canceled.

Note:

Use this argument at the owning site only.

cancel_replica_co (crepco)

Cancels replica checkouts based on the specified user ID, group name, and site name. Canceling a replica checkout also cancels the remote checkout at the owning site.

If no user, group, or site is specified, all replica checkouts are canceled.

Note:

Use this argument at the owning site only.

remote_import (ri)

Imports the item specified by the **-item_id** argument from the owning site or the site specified by the **-site** argument. If the item is a replica at the local site, it is imported from the owning site and any site specified in the command is ignored.

When you specify the **-optionset** argument, Multi-Site uses a TC XML payload to exchange data. For 4GD data, you must specify the **-optionset** argument.

Note:

Wildcard characters cannot be used in the **-item_id** argument.

You can also use this argument to import a list of items from an input file designated by the **-filename** argument. The input file must contain UIDs for the items to be imported, and the **-classoffile** argument value must be set to **Tagstring** when using an input file. You can use the **sync_on_demand** utility to generate a file that contains UIDs of items of an assembly enclosed within square brackets ([]) or other designated separator. You can then write a script to collect the UIDs into the input file.

The **data_share** arguments related to variants and line of usage (LOU) cannot be used with the **remote_import** argument.

By default, items are processed sequentially. Use **-bulk** with **-f=remote_import** to process the items in a single operation.

offline_export

Exports the objects to the directory on the file system specified by **-dir**.

-meta_file

Generates only metadata files without exporting objects. Used only with **offline_export** and **offline_import**. Typically used for data migration.

offline_import

Used with **-dir**, imports the objects to the directory on the file system.

-bypass_site_check

Import metadata file to a site other than the original target site. Used only with **offline_import**.

-meta_file

Generates only metadata files without importing objects. Used only with **offline_export** and **offline_import**. Typically used for data migration.

register_migration_data

Registers the Teamcenter objects to be migrated between different Teamcenter releases. See Migrating Teamcenter data to a newer version of Teamcenter.

install_http_proxy_user

Installs a proxy user at the site specified with **-site** or update the details of the proxy user at the site specified with **-site**. With **install_http_proxy_user**, use the following arguments to specify the proxy user details:

-proxy_user=*proxy_user_name*

-proxy_group=*proxy_user_group*

-proxy_role=*proxy_user_role*

-proxy_user_pwd=*proxy_user_password*

See Configure Multi-Site authentication using secondary LDAP servers configured with TcSS.

remove_http_proxy_user

Uninstalls the proxy user at the site specified with **-site**. See Configure Multi-Site authentication using secondary LDAP servers configured with TcSS.

-dir

Specifies the local directory in or from which objects are imported or exported. Required with **-f=offline_export** and **-f=offline_import**.

-site

Specifies the name of the site to which objects are published or from which they are unpublished. It can be given multiple times in a command line.

-owning_user (ou)

Specifies the user ID of the user at the remote sites to which the objects are sent. The specified user owns the objects being sent. See *Restrictions*.

-owning_group (og)

Specifies the name of the group at the remote sites to which the objects are sent. The group owns the objects being sent. See *Restrictions*.

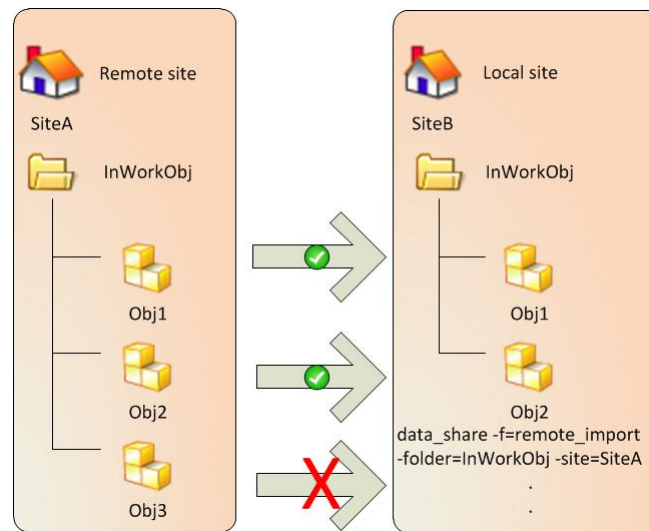
-item_id (item)

Specifies the item ID or template of items to process. It is mutually exclusive with the **-folder**, **-filename**, **-keyFileName**, **-name** and **-key** arguments. It is required for the **-delete_pubrec** argument.

-folder (fl)

Specifies the name of a Teamcenter folder containing the list of objects to process. It is mutually exclusive with the **-name**, **-filename**, and **-item_id** arguments.

During a remote import, if the named folder does not exist at the replica site, the utility generates an error message. If the folder is empty at the replica site, the utility does not pull any objects from the remote site. If the folder exists and has content, the objects that it contains are imported from the remote site.



Note:

If the **-include_bom** argument is used with the **-folder** argument, only the **ItemRevision** objects (and objects related to them) in the folder are replicated. The folder itself is not replicated.

-name

Specifies the name of a single workspace object to be precessed. If not an item, use the **-class** argument to specify the class of the object. It is mutually exclusive with the **-folder**, **-filename**, and **-item_id** arguments.

For organization objects, this argument accepts the following attributes:

- **User** objects require a **user_ID** attribute.
- **Role** objects require a **role_name** attribute.
- **Group** objects require the groups full name that uniquely identifies the group.

- **Person** objects require a **user_name** attribute.

Note:

You can use a text file containing a list of all organization objects of the same type that you want to export using the **-classoffile** and **-filename** arguments.

-filename (fn)

Specifies the name of the input file containing the list of IDs or names of objects to process. File entries are treated as IDs for **Items** and **ItemRevisions** objects and as names for other classes of objects. It is mutually exclusive with the **-name**, **-folder**, **-item_id**, **-key**, **-itemKeysFile**, and **-itemRevisionKeysFile** arguments.

If the input file contains names, the **-classoffile** argument is required.

-key

Specifies the keys of the items to process, the template of the item keys, or the 4GD object key. It is mutually exclusive with the **-item_id** argument. Use the following format:

```
keyAttr1=keyVal1,keyAttr2=keyVal2...,keyAttrN=keyValN
```

To find the key of an object, use the **get_key_string** utility.

-itemKeysFile

Specifies the name of the input file containing the list of item key strings of the items you want to update. The following listing shows sample content of a file for updating a list of items:

```
item_id=export_001
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

If the item key file has 4GD object key strings, the corresponding 4GD class must be supplied using the **-classoffile(cof)=** argument. The following listing shows sample content of a file for updating 4GD data:

```
4gd_id=DE_Export_001
4gd_id=DE00001_ID,object_name=DE00001_Name,object_desc=DE00001_Desc
```

The **4g_id** entry maps to the corresponding unique ID of the 4GD class, for example:

```
Class=Cpd0DesignElement, 4gd_id=cpd0_design_element_id
```

-itemRevisionKeysFile

Specifies the name of the file containing the keys of the item revisions to process. It is mutually exclusive with the **-item_id** argument.

-class (cl)

Specifies the Teamcenter class of the object specified by the **-name**, **-key**, or **-4gd_id** argument. This argument is valid only with the **-name** or, when the **-optionset** argument is specified, with the **-4gd_id** argument. The default class is **Item**.

For organization objects, this argument accepts **Role**, **User**, **Group**, and **Person** classes.

-classoffile (cof)

Specifies the class of the objects listed in the input text file given with the **-filename** argument. If not defined, the default class is **Item**. It is required if the input file has names instead of IDs.

For organization objects, this argument accepts **Role**, **User**, **Group**, and **Person** classes.

-include

Specifies a relation type to include. This can be specified multiple times in a command line. The database name (not the display name) of the relation type must be used.

For best TC XML performance, use a closure rule for relation instead of include/exclude.

-exclude

Specifies a relation type to be excluded from the operation. This can be specified multiple times in a command line. The database name (not the display name) of the relation type must be used.

The list of relation types to be included is determined by either the **TC_relation_required_on_export** (export without transferring ownership) or **TC_relation_required_on_transfer** (export with transfer of ownership) preferences.

The **IMAN_master_form** relation cannot be used as an argument for the **-exclude** argument. The following relations cannot be used as an argument for the **-exclude** argument unless they are not included as a value in the preference for the location:

- **IMAN_requirement**
- **IMAN_specification**

For best TC XML performance, use a closure rule for relation instead of include/exclude.

-revision-selector

Identifies the item revisions to send. It is also used as the revision rule for identifying components when processing assemblies. When used with the **-include_bom** argument while publishing or unpublishing, it determines which revisions' BVR to follow in traversing the assembly tree.

If no revision selector is specified, the default selector is **all_revisions**.

The valid revision selectors are as follows:

all_revisions

Sends all revisions. it is not valid when publishing.

latest_revision

Processes only the latest revision regardless of release status. This is the default if no revision selector is given when publishing or unpublishing.

selected_revision

Process only the selected revision.

latest_working

Processes only the latest working revision.

latest_released

Processes only the latest released revision with any release status.

latest_working_or_any

Sends only the latest working revision. If none, the latest released revision is processed.

release_status

Processes only the latest released revision with the given release status.

all_released_revs

Sends all revisions with a release status; it is not valid when publishing.

-rev

Specifies the ID of a specific item revision to be sent to a remote site. It is valid only with the **-item_id** argument and with the **-send** function, and is mutually exclusive with revision selectors.

You can use the **-rev** and **-include_bom** arguments together to send a precise assembly by including the **-assert_precise** argument.

For 4GD content, you can set this argument as **-rev=baseline_rev** along with the following arguments set as **-4gd=baseline_id** and **-class=MdloBaseline** to select specific 4GD baseline content.

-include_bom (bom)

Includes assembly components when sending, publishing, or unpublishing. A revision selector is required when publishing or unpublishing an assembly; if no revision selector is given, the **latest_revision** selector is used as the default. When sending, the default selector is **all_revisions**.

If not specified, this argument defaults to **off**.

This argument does not traverse the component relationships of subassemblies. This allows you to send the subassemblies in separate transactions. Using multiple, simultaneous transactions to transfer very large assemblies and their subassemblies separately provides improved scalability and performance.

Note:

This argument, although similar to the rich client **Include Entire BOM** remote export option, may not export the same set of objects. The **Include Entire BOM** option traverses all components, subassemblies, and subassembly component relationships that can result in unacceptable performance for very large assemblies.

-include_modified_only

Includes only modified item objects, resulting in improved performance. For example, if an item has revisions A and B, where B is the new revision, only revision B and its dataset are sent to the remote site.

-assert_precise (ap)

Specifies that the assembly is precise. The **-rev** and **-include_bom** arguments can be used together for precise assemblies only. You must include this argument to send a specific revision of an assembly to a remote site.

-transfer (tf)

Transfers site ownership when sending objects. Site ownership is not transferred by default.

-attach (att)

Attaches an object to the appropriate parent item or revision at the receiving site when sending an attachment with transfer of site ownership. Use this for situations in which you attach a dataset to a replica, such as a JT file, and you want to send the JT file to the owning site with transfer of site ownership and attached to the appropriate parent item or revision.

-exclude_files (exf)

Excludes dataset files.

-latest_ds_version (ldv)

Sends only the latest dataset version. Unless this argument is specified, all dataset versions are sent.

-exclude_folder_contents (efc)

Excludes the contents of folders being exported.

-include_bc (ibc)

Identifies the **BomChange** objects associated with the affected assemblies to send. If not specified, **BomChange** objects are not sent.

-include_supercedures (isc)

Identifies the supercedure objects associated with the **BomChange** objects to send. If not specified, supercedure objects are not sent.

-include_pfmembers

Identifies the related part family members to be exported when handling part family templates.

-include_pftemplates

Identifies the related part family template to be exported when handling part family members.

-pf_bom_treatment

Identifies the part family objects associated with the assemblies to be exported. The argument must be used in conjunction with the **-include_bom** argument. Valid arguments are:

-members

Includes part family member components present in the assembly.

-templates

Includes part family template rather than part family member components.

-all

Includes both the part family member components and templates.

-none

Includes neither the part family member components nor the templates.

-qry-name

Runs a saved Query Builder that defines the list of objects to process. One or more **-qry_attr** and **-qry_val** pairs specify the attribute value pairs the objects must match to be processed.

-oaat

Forces the utility to process one object at a time. Normally, this utility processes all objects in a batch. If there is a failure on any top-level object, the entire batch fails. With this option, the other objects are successfully processed when one object fails. The log file indicates which object failed. The disadvantage to using this argument is the utility takes longer to process objects individually.

This argument is valid only if you specify the **send** function.

-continue_on_error (con)

Specifies processing is continue if there is an error on an optional object such as a reference or manifestation. This argument is not valid when transferring site ownership.

Outputs the error in a report file and continues processing the other items. The **-report** argument must be specified to see the error.

-batch_size (bs)

Specifies the number of objects per batch; a new process is created per batch. The default batch size is 1000. The value must be a positive integer. **-batch_size** is useful when processing thousands of objects, because it helps avoid memory and disk space shortage problems.

-report (rep)

Specifies to output a report to the specified file.

-user

Specifies the user ID.

-group

Specifies the group name.

-error_file (err)

Specifies the name of the output error report when sending assemblies.

-mapping_rule

Specifies the name of the `.xsl` file detailing the object transformation rules needed when transferring objects to a site with a different schema. You can create this `.xsl` file by running the **plmxml_tm_edit_xsl** utility with the **-action=export** argument on the transfer mode on which the transformation rules are attached.

You can specify mapping transfer modes for Advanced Multi-Schema Exchanger to use for Multi-Site TC XML importing and exporting using the **TC_tms_site_interop_transfer_mode** and **TC_tms_export_mapping_transfer_mode** preferences. These preferences do not affect Briefcase and PLM XML importing and exporting with Advanced Multi-Schema Exchanger.

-exclude_variant_options (evo)

Indicates all variant options are to be excluded during a send operation.

-batch_variant_options (bvo)

Indicates all variant options are sent separately in batch mode.

-batch_objects (bo)

Indicates all objects of the given classes are sent separately in batch mode. Separate each class name with a comma. The list cannot contain spaces. The following table is a list of supported classes for this argument:

Dataset	ImanRelation	PSOccurrence
Folder	MEAppearancePathNode	VariantExpression
Form	NamedVariantExpression	VariantExpressionBILock
	(Not supported by Multi-Site Collaboration.)	(Not supported by Multi-Site Collaboration.)

-batch_file (bof)

Indicates all objects of the classes in the specified file are sent separately in batch mode. List each class name separately on a line in the file. For a list of supported classes for this argument, see the table for the description of the **batch_objects** argument.

-include_dist_comp

Includes distributed components during import. This argument is valid only in conjunction with the **-remote_import** argument.

-lightweightobject (lwo)

Send the specified lightweight object unique identifier in the specified file. For example,

```
data_share -f=send -site=site2 -filename=uid.txt -
classoffile=Tagstring -lwo=ImanExportRecord
```

-log

Specifies detailed log information is written to the log file.

-checkpoint (cp)

Initiates a checkpoint transaction; that is, a transaction that can be restarted at the point of failing.

This argument is valid only with **send** function. It is not valid with the **-transfer** argument.

If a noncheckpoint operation is initiated for multiple target sites and some target sites are not currently available based on a preliminary availability check, Teamcenter sends a message to **stdout** to notify the user about unavailable sites, removes unavailable sites from the target site list, and then performs the operation for the available sites.

-compress_ind_files (cif)

Specifies compression mode to use to compress files in the export directory during a checkpoint transaction. If not specified, it creates a single large ZIP file. This argument is valid only with the **-checkpoint** argument. Valid values are:

S	Creates a single large ZIP file.
I	Creates a ZIP file for each individual file, resulting in multiple ZIP files.
N	No files are compressed.

-transaction_id (trid)

Specifies a 14-character transaction ID for a given checkpoint-related operation or for a fast sync transaction if used with the **-optionset** argument. A fast sync transaction provides improved performance when synchronizing data using the **data_sync** utility.

-tcx_data_load

Improves performance by disabling Multi-Site Collaboration business logic such as AM rule checks and in-process workflow data checks. This argument is intended only for migration scenarios.

Warning:

Use **-tcx_data_load** only in controlled migration scenarios. Disabling business logic in standard operation scenarios may cause unexpected changes in your data.

-tcx_data_load can be used with the **send**, **remote_import**, and **offline_export** functions.

-migration_sync

Reports unsynchronized data in support of migration scenarios. Used with **-tcx_data_load**, for example,

```
data_share -migration_sync -tcx_data_load -since=YYYY-MM-DD:HH:NN -
f=offline_export -dir=c:\outofsync
```

-status (stat)

Displays the status of a given transaction ID.

If the **-site** switch is given, only status of the given sites are displayed. If no **-site** switch is given, the status returned depends on whether the command is given at the site that initiated the checkpointed transaction or the site is the receiving end of the transaction. If the command is given at the initiating site and no **-site** switch is given, the status of the local site and all target sites is returned.

-cleanup_transaction (ct)

Removes transient data generated during a checkpoint transaction or fast synchronization transactions if used with the **-optionset** argument.

For checkpoint transactions, the transient data consists of the export data and supporting directories and files used to manage the transaction. You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have **delete** access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an error message is displayed to **stdout** and the utility returns a nonzero value.

For fast sync transactions, you must specify either the **-transaction_id** or **-before_last_process_date** argument.

-before_last_process_date (blpd)

Specifies the date used to determine which fast sync transactions to clean up. The date must be supplied in the following format:

```
YYYY-MM-DD:HH:NN:SS
```

YYYY represents the four-digit year value. MM represents the two-digit month value. DD represents the two-digit day of the month. HH represents a two-digit hour value from 0 to 23. NN represents a two-digit minute value from 0 to 59 and SS represents a two-digit second value from 0 to 59. The DD:HH:NN:SS are optional. If they are not specified, the utility uses 12:00 AM of the specified date.

Valid only with the **-cleanup_transaction** argument.

-restart (rs)

Restarts a given transaction at the point of failure.

Valid only with the **-f=send** function.

-commit_ixr (cmi)

Updates the export records at the owning site once the data is known to have been successfully imported at a target site.

Under normal conditions, the update of the export records are performed automatically by each subprocess that succeeds in completing the send operation to its assigned site. Use this function only if either of the following conditions occur:

- The failure occurs at the importing site, and the user performs the restart using the **item_import** utility.
- The failure occurs after the data is successfully imported by a target site, but a failure occurs just before or during the updating of the export records.

You must use at least one **-site** argument to identify the site or sites for which export records are to be updated.

You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have **read** access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an error message is displayed to **stdout** and the utility returns a nonzero value.

-list_transactions (lt)

Lists all uncleaned checkpoint transactions or fast sync transactions if used with the **-optionset** argument. An uncleaned transaction is one in which its transient data has not been deleted from the transfer area using the **cleanup_transaction** function.

- Active transactions can only be detected at the site that initiated it. The receiving end of a transaction is not able to determine if a transaction is active or not.
- The list of inactive transactions initiated by the local site and the list of transactions initiated by remote sites are based only on the contents of the transfer area of the node where this command is executed.
- The list of active transactions initiated by the local site is always complete because it is based on data stored in the local database.

You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have **read** access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an appropriate error message is displayed to **stdout** and the utility returns a nonzero value.

-all_subgroups

Exports all subgroups of the selected group. Parent groups are always exported. This argument is valid only when the **-class** or **-classoffile** argument is set to **Group**.

-all_roles

Exports all roles associated with the selected group. All roles for subgroups are included if the **-all_subgroups** argument is also specified. If not specified, only the default role is exported.

This argument is valid only when the **-class** or **-classoffile** argument is set to **Group**.

-all_groupmembers

When exporting a **User** object, exports all **GroupMember** objects related to the selected user. When exporting groups or groups and subgroups (**-all_subgroups** argument), exports all **GroupMember** objects related to any role that is exported.

You must specify one of the following arguments to use this argument:

```
-class=User
-classoffile=User
-class=Group
-classoffile=Group
```

If this argument is not specified when exporting **User** class objects, only the default group related to the user is exported.

If this argument is not specified when exporting **Group** class objects, no **GroupMember** objects are exported.

-optionset

Exports the data using the TC XML functionality. This method provides better performance for large data transfers and must be used when exporting 4GD data.

You can replicate 4GD objects to a remote site when you specify the **-optionset** argument with the **send** function. Optionally specify the transfer option set used for the export by setting **-optionset** to the set name. If you do not specify an option set, the utility uses **MultiSiteOptSet** as the default transfer option set value. Values of the options listed in the option set govern the object export. The option set must exist at the exporting site.

For 4GD objects, you can also specify the **-de_incl_rlz_bom** and **-workset_incl_relz_de** arguments.

-de_incl_rlz_bom

Exports the source objects of a design element (Type:**Cpd0DesignElement**). You must specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument. You must specify the **-optionset** argument and specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument.

-workset_incl_relz_de

Exports the source objects of a design element (**Cpd0DesignElement**) in a workset (**Cpd0Workset**). You must specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument. You must specify the **-optionset** argument and specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument.

-4gd_id

Specifies a 4GD object identifier or 4GD object pattern. The **-class=class-name** argument must be specified with the **-4gd_id** argument.

The utility maps the **-4gd_id** argument to the corresponding unique ID of the 4GD class, for example:

```
Class=Cpd0DesignElement, 4gd_id=cpd0design_element_id
```

A 4GD partition object and 4GD subset definition objects do not have a unique 4GD class ID. Therefore, using **-4gd_id** for partition objects or subset definition objects may result in transfer of multiple objects.

To export unique partition object use multifield key attributes supplied in the **-key** argument, see [Examples](#).

-parallelize

Specifies that parallel processing is enabled at the owning site and remote site when replicating data. **-parallelize** is valid only with the **-optionset** and **-send** functions. **-parallelize** is not valid with **-transfer**.

The number of additional to use processes at each site is defined by setting **-parallelize** to two comma-separated values. The first value specifies the number of additional processes to use at the owning site. The second value specifies the number of additional processes to use on the remote site. For example, to specify 5 additional processes on the owning site and 7 additional processes on the remote site, set **-parallelize** to **5,7**.

-parallelize supports replication to a single site. Ownership transfer is not supported.

Note:

Be judicious when specifying the number of additional processes to use at each site. If the workstation is powerful, with unused CPU cores and RAM, using parallel processes can improve performance. However, specifying parallel processing on less powerful workstations may overtax the workstations, draining system resources.

-override_options

Override the following default operations by setting the following values to **true**. (The default value of each is **false**.) Options and values are separated by colons. Multiple overrides are separated by commas.

opt_exclude_apn When exporting BOM structures, set to **true** to exclude APNs from being exported.

opt_include_rendering Set to **true** to include the rendering dataset in the exported data.

opt_include_incremental_change_element When exporting **EngChange** lists, set to **true** to export **IncrementalChangeElement** items.

opt_exp_all_wso Set to **true** to export secondary objects for all relations. Use with care as this may result in a very large amount of data being exported.

-session_options

Override the following default session operations by setting the following values to **true**. (The default value of each is **false**.) Options and values are separated by colons. Multiple overrides are separated by commas.

forceUpdate Set to **true** to force an update during an import if a replica's last saved date (**lsd**)
OnImport value or last modified date (**lmd**) value is greater than the primary object's value.

opt_traverse Set to **true** to export only an island of data. Do not export across islands.
_by_island

-volume_file_list

Generates a text file listing the volume file paths associated with successfully imported or exported items. This information is particularly useful when migrating data. Used with **-offline_import** and **-meta_file**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

1. When sending objects to a specific user and/or group at a remote site using the **-owning_user** and **-owning_group** arguments, the following rules apply:
 - To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
 - If both the specified user and group exist at the importing site, the imported objects are owned by the user and group regardless of whether or not the user is a member of the group.
 - If only the user is specified or if the group is specified but does not exist at the importing site, the user's default group at the importing site is the owning group of the imported objects.
 - If only the group is specified or if the user is specified but does not exist at the importing site, the user context of the remote IDSM process is the owning user of the imported objects.
2. If variant options are excluded using the **-exclude_variant_options** argument, it is implied that they cannot be sent separately in batch mode. Therefore, the **-exclude_variant_options** argument

cannot be used with either **-batch_variant_options**, **-batch_objects=variant-expression**, or with the **-batch_file** arguments when the given file includes the class name **VariantExpression**.

3. Any number of objects can be sent separately in batch mode. Class names of objects can be given in a comma-delimited list with the **-batch_objects** argument or listed in a file whose name is specified in the **-batch_file** argument.
4. The **-item_id**, **-name**, **-filename**, **-folder**, **-key**, **-itemKeyFile**, **-itemRevisionKeysFile**, and **-4gd_id** arguments are mutually exclusive
5. Traversal-free synchronization is not supported for remote import (**-f=remote_import**).
6. If the replica does not exist at the importing site, only the **-4gd_id** and **-item_id** arguments support the remote import function. (**-f=remote_import**).

For objects specified by **-filename**, multifold **-key**, **-name**, or **-class** arguments, replicas must exist on the importing site.

All arguments supported for the send function (**-f=send**) with the **-optionset** argument are supported for remote import.

EXAMPLES

Required log-in information is omitted from the following examples.

- To send a list of items specified in a text file to two sites:

```
data_share -f=send -filename=my_item_list.txt -site=Site1 -site=Site2
```

- To send a list of items specified in a text file and output a report; continue processing if a nonfatal error is found:

```
data_share -f=send -filename=my_list.txt -site=Site1
-report=rep.txt -coe
```

- To send a precise assembly to a remote site:

```
data_share -f=send -item_id=xyz -rev=A -include_bom
-assert_precise
```

- To transfer ownership of a given item:

```
data_share -f=send -transfer -item_id=item123 -site=Site1
```

- To publish an assembly item and all its components using the latest revision rule to determine components:

```
data_share -f=publish -item_id=Engine100 -site=Ods1 -include_bom
```

- To publish an assembly item and all its components using the latest released revision rule to determine components:

```
data_share -f=publish -item_id=Item1 -site=Ods1 -include_bom  
-latest_released
```

- To unpublish an assembly item and all its components from multiple ODS sites using the default revision rule **latest revision**:

```
data_share -f=unpublish -item_id=Item1 -site=Ods1 -site=Ods2  
-include_bom
```

- To delete a publication record in the local database:

Note:

Use this only if the primary object has been deleted, but the publication record still exists.

```
data_share -f=delete_pubrec -item_id=ObsoleteItem1
```

- To list the authorized ODS sites:

```
data_share -f=list_ods
```

- To check availability of the authorized ODS sites:

```
data_share -f=check_ods
```

- To get publication information about a list of objects in a folder:

```
data_share -f=list_pub_info -folder=myFolder
```

- To send an item to a specific remote user and group:

```
data_share -f=send -item_id=xyz -site=Site1 -owning_user=joe  
-owning_group=engg
```

- To send an item to a site using a different schema. In this scenario, Advanced Multi-Schema Exchanger is used to create the transformation rules to use when exchanging files. After attaching the rules to the transfer mode, the rules are extracted to a .xsl file using the **plmxml_tm_edit_xsl** utility with the **-action=export** argument on the transfer mode.

```
data_share -f=send -item_id=0001 site=site2  
-mapping_rule=c:\mapping.xslt
```

- When publishing thousands of items and you get errors after publishing several hundreds or even thousands of items, reduce the batch size:

```
data_share -f=publish -item_id=A* -site=Site1 -batch_size=200
```

- To publish an engineering change object and all its associated change objects:

```
data_share -f=publish -item_id=CR0001 -site=Ods1 -include_bom  
0-include_bc -include_supercedures
```

- To register an item ID with the central item ID registry:

```
data_share -f=register -item_id=myItem
```

- To find duplicate item IDs at another site:

```
data_share -f=find_duplicates -item_id=00* -site=Site1
```

- To list all objects that are checked out by remote users:

```
data_share -f=list_remote_co
```

- To list all objects that are checked out by user **justin** at **Site2**:

```
data_share -f=list_remote_co -user=justin -site=Site2
```

- To cancel check out of all objects by user **joseph** at **Site2**:

```
data_share -f=cancel_remote_co -user=joseph -site=Site2
```

- To list all replica objects that are checked out by local group **engg** from remote site **Site1**:

```
data_share -f=list_replica_co -site=Site1
```

- To cancel all replica objects that are checked out by user **davis** from **Site1**:

```
data_share -f=cancel_replica_co -user=davis -site=Site1
```

- To cancel remote checkout on a given item:

```
data_share -f=cancel_remote_co -item_id=item123
```

- To cancel remote checkouts on the datasets listed in the **dataset.lst** file:

```
data_share -f=cancel_remote_co -filename=dataset.lst -class=Dataset
```

- To cancel replica checkouts on the datasets in uniquely named folder:

```
data_share -f=cancel_replica_co -folder=unique_folder_xyz
```

- To exclude all variant options during a send operation:

```
data_share -f=send -item_id=CR0002 -site=remote1
-exclude_variant_options
```

- To batch send all variant options during a send operation:

```
data_share -f=send -item_id=CR0002 -site=remote1
-batch_variant_options -batch_size=10000
```

- To batch send one or more classes of objects during a send operation:

```
data_share -f=send -item_id=CR0002 -site=remote1
-batch_objects=class1,class2 -batch_size=10000
```

- To batch send one or more classes of objects given in a text file during a send operation:

```
data_share -f=send -item_id=CR0002 -site=remote1
-batch_file=my_list.txt -batch_size=10000
```

- To import an item from a remote site (**Site2**):

```
data_share -f=remote_import -site=Site2 -item_id=xyz
```

- To import the objects of an assembly using a file (**UIDs_list.txt**) containing a list of UIDs:

```
data_share -f=ri -filename=UIDs_list.txt -classoffile=Tagstring
```

- To initiate a checkpoint transaction at three specified sites:

```
data_share -f=send -checkpoint -item_id=item123
-site=Site2 -site=Site3 -site=Site4
```

- To export a role that does not have any group members from the **TopGrp1** group:

```
data_share -class=group -name=TopGrp1 -f=send
-all_groupmembers -site=Site2
```

- Include the rendering dataset and **IncrementalChangeElement** items in the exported data:

```
data_share -f=send -item_id=xxx -site=site2
-override_options=opt_incl_rendering:true,opt_include_ice:true
```

- Force updating of the data on the import site even though the last saved dates are the same for the primary and replica:

```
data_share -f=send -item_id=xxx -site=site2
           -session_options=ForceUpdateOnImport:true
```

- To return status for a transaction ID of **AhEZaOnRAAMfD** and no **-site** argument is specified:

```
data_share -f=status -trid=AhEZaOnRAAMfD
```

The output is similar to the following:

```
Site1: sending export data to all target sites (06-Nov-2007.14:31:28)
Site2: transmitting data (06-Nov-2007.14:35:31)
Site3: importing data (06-Nov-2007.14:34:29)
Site4: error 100107 - site not currently available (06-Nov-2007.14:32:26)
Site5: transaction complete (06-Nov-2007.14:40:10)
```

The time stamp represents the last time the status was updated.

- To return status at a receiving site and no **-site** argument is specified:

```
Site3: importing batch 5 out of 50 (06-Nov-2007.14:34:29)
```

At a receiving site, only the status of the local site is obtained; the status of other sites involved in a transaction are not available.

- To restart a given transaction for a given site:

```
data_share -f=send -transaction_id=AhEZaOnRAAMfD
           -restart -site=Site3
```

- To update export records at site **Site4** using a transaction ID of **AhEZaOnRAAMfD**:

```
data_share -f=commit_ixr -trid=AhEZaOnRAAMfD -site=Site4
```

- To clean up records with a transaction ID of **AhEZaOnRAAMfD**:

```
data_share -f=cleanup_transaction -trid=AhEZaOnRAAMfD
```

- To list checkpoint transactions:

```
data_share -f=list_transactions
```

The output is similar to the following:

Transactions initiated by local site:

AhEZaOnRAAMfD - active

BxyzZaOnRAAXYZ - inactive

Transactions initiated by remote sites:

ZaOnRAAYXCDA - Site3

- To send the **casCD001** collaborative design object to the **site2** site:

```
data_share -f=send -4gd_id=casCD001 -class=Cpd0CollaborativeDesign
-site=site2
```

- To transfer architecture breakdown structures between sites, you must execute the following steps sequentially:

1. Push the design assembly.

```
data_share -f=send -item_id=design-assembly-item_id -site=remote-site-id
-exclude=IMAN_reference -exclude=IMAN_based_on -exclude=IMAN_snapshot
-exclude=IMAN_3D_snap_shot -exclude=IMAN_external_object_link
-exclude=TC_Generic_Architecture -bo=VariantExpression, MEAppearancePathNode
-bs=1000 -ldv
```

2. Push the **LOUHOLDER** object and synchronize the BOM view revision.

```
data_share -f=send -item_id=<ITEM_ID> -site=<REMOTE_SITE> -include_bom
-exclude=IMAN_reference -exclude=IMAN_based_on -exclude=IMAN_snapshot
-bvo -bo=PS0ccurrence -bs=20 -bvrsync -report=<IMAN_TMP_DIR>/rpt
```

3. Push the architecture breakdown structure excluding the NVEs.

```
data_share -f=send -item_id=<ITEM_ID> -rev=001 -site=<REMOTE_SITE>
-exclude=IMAN_3D_snap_shot -exclude=IMAN_reference
-exclude=IMAN_external_object_link
-exclude=IMAN_based_on -evo -bo=MEAppearancePathNode -bs=2000 -ldv
```

4. Push the NVEs.

```
data_share -f=send -item_id=<ITEM_ID> -rev=001 -site=<REMOTE_SITE>
-exclude=IMAN_3D_snap_shot -exclude=IMAN_external_object_link
-exclude=IMAN_based_on
-bo=MEAppearancePathNode -bs=2000 -ldv
```

- To send a 4GD object and output a report:

```
data_share -f=send -4gd_id=Ste2_DE0524 -cof=Cpd0DesignElement
-site=Site1 -optionset -report=4GD_rep.txt
```

- To send a list of 4GD objects specified in a text file using parallel processing and output a report:

```
data_share -f=send -filename=my_4GD_list.txt -cof=Cpd0DesignElement
-site=Site1 -optionset -parallelize=6,8 -report=rep.txt
```

- To send a specific 4GD partition object and output a report:

```
data_share -f=send class=Cpd0DesignElement
-key=ptn0partition_id=CD001_id,ptn0partition_scheme_type=SchemeFunctional_CD001,
partitionptn0source_object=partition_source_CD001,mdl0model_object=model_CD001,
mdl0revision_id=model_id=001/A site=Site1 -optionset -report=rep.txt
```

- To clean up fast synchronization transactions prior to specific last process date, list the available transactions to get the last process dates:

```
data_sync -optionset -lt
```

Clean up the transactions:

```
data_sync -optionset -ct -blpd=2012-12-18:20:10:00
```

- To use the **TC_cms_relation_optset_map** preference to include or exclude relations:
 1. Add the relation and option set to the **TC_cms_relation_optset_map** preference, for example:

```
IMAN_rendering, opt_rel_rendering
```

2. In the PLM XML/TC XML Export Import Administration application, expand the **TransferOptionSet**, click **MultiSiteOptSet**, and add the **opt_rel_rendering** option with the default value set to **false**.
3. Expand **ClosureRule**, click **MultiSiteDefaultCR**, and add the following clause:

```
CLASS:WorkspaceObject:CLASS:Dataset:RelationP2S:IMAN_rendering:
SKIP:$opt_rel_rendering==false;
```

This clause states, from any **WorkspaceObject**, find the dataset using **IMAN_rendering** relation, and when the relation is **opt_rel_rendering**, skip the dataset during export. It means the default is to always exclude the **IMAN_rendering** relation for an exported object.

4. To include the **IMAN_rendering** in the export using the **data_share** utility, type:

```
data_share -include=IMAN_rendering -optionset
```

data_sync

Synchronizes copies of objects at remote sites with the latest version of the primary object. It also updates publication records when republishing objects. In **verify** mode, the utility checks the existence of exported objects at the remote sites; if a copy no longer exists at the remote site, the corresponding import export record is deleted from the owning site.

The behavior of this utility is controlled by the **TC_force_remote_sites_exclude_files** preference. If this preference is set to **true**, the replica files stored in the remote site file server cache (FSC), otherwise the replica files are store in the remote system volume.

The **data_sync** utility uses import export records (IXRs) and publication audit records (PARs), which are attached to the primary copy of an object, to determine whether or not to synchronize a copy or the publication record in the ODS. These records contain information on when the object was last sent to a particular site or last published to an ODS. It then compares these dates with the object's last-modified date and decides whether or not to synchronize the object. Thus, only those objects that were modified since the last successful run of the utility are updated.

When updating multiple sites and not all sites are operational, the **data_sync** utility updates the sites that are available but remembers, using the IXRs and the PARs, which ones were unavailable so they can be updated next time.

Once the utility determines which objects and sites to synchronize, it uses the basic Multi-Site Collaboration mechanisms (export, import, IDSM, and ODS) to accomplish its task. For this reason, Siemens Digital Industries Software recommends that the **data_sync** utility be run in batch mode during off hours so that it does not compete for computing and network resources during business hours.

The utility also supports TC XML transfers for 4th Generation Design (4GD) data. The 4GD relation data mapping is controlled by the **TC_cms_relation_optset_map** preference. You use this preference when you want to control the relations that are included or excluded when a 4GD object is replicated.

The **data_sync** utility supports part family templates and members. It also supports organization classes, specifically, **Role**, **User**, **Group**, and **Person** classes.

Siemens Digital Industries Software recommends the following practices when using the **data_sync** utility. The term *one at a time* means one command line invocation. This implies that your script for running **data_sync** consists of several lines invoking the **data_sync** utility.

- Synchronize one site at a time and use the default revision selector of **-same_as_last_export**. This allows you to use the Smart Sync capability which synchronizes only the revisions and attachments that the remote user specified when replicating an item.
- Synchronize one class at a time starting with the largest unit, which is **Item**, and down to the smallest units such as **Dataset** and **Form**.
- Always use the **-since** switch with the **-class** switch. This results in improved memory efficiency because replicated objects that have not been modified for some time are excluded from the initial

search for objects to be synchronized. Ideally, the date given to the **-since** switch should be the exact date and time of the last successful run of **data_sync**. However, if you are not sure about the date and time, use a date and time that you know is prior to the last successful run.

- When dealing with thousands of objects, **data_sync** tends to slow down as it loads more and more objects in memory. It handles this problem by cascading its work over several sub-processes. When the original process reaches its batch size, it starts another process and then terminates itself; the sub-process continues where its parent process left off. When it reaches its own batch size, it creates another sub-process, and so on. The optimum batch size varies for each installation depending primarily on the memory (both main memory and virtual memory), so you must determine the optimum batch size for your installation.

One tool that can help you do this is the use of the **-log** switch that records all significant events in the **data_sync** log file, the file with the **.log** extension. By analyzing the log file, you can detect at what point **data_sync** begins to slow down so you can then adjust the batch size accordingly. Note that the use of the **-log** switch itself can affect the overall efficiency of **data_sync** so you should turn off the switch once you have determined your optimum batch size.

- The synchronization process can put a heavy load on the network and the systems so **data_sync** should be scheduled during non-busy hours such as nights and weekends. Typically, you should run the synchronization script as a cron job to be started at night.
- It is not necessary to have a separate verify run before synchronization because **data_sync** always performs a verification before synchronization. View a separate verification run as a cleanup procedure and run it only when the network and the systems are not busy, such as on weekends.
- Do not use the **-disable_modified_only** switch unless there is a known problem with the default synchronize modified objects only mode.
- If you typically share whole assemblies with a site, it is best to use the **-filename** switch to synchronize specific assemblies and use the **-include_bom** switch to synchronize any modified components. Note that you may have to use the **-force** switch in the event the item itself was not modified but you want to synchronize modified components.

The behavior of the utility for project relationships on replica objects can be controlled by the **TC_sync_projects_with_owning_site** preference. This preference is not created by the Teamcenter installation process. To change the default behavior of shared project relationships, you must create the preference.

AVOID OUT-OF-MEMORY ERRORS

To avoid possible out-of-memory errors when you are replicating architecture revisions, Siemens Digital Industries Software recommends you exclude **MEApperancePathNode** (APN) objects by setting the following environment variable:

```
TC_EXCLUDE_APN=TRUE
```

This excludes APN objects from the synchronization. It may also exclude associated object, such as JT for promoted and deformed bodies. You can synchronize the APN objects by using the **sync_product_apns** utility.

If you do not set this environment variable, or you set its value to **FALSE**, you can use the batch feature of this utility to overcome memory limitations when synchronizing large numbers of APNs. For example, you may specify:

```
-batch_objects=MEAppearancePathNode -batch_size=5000
```

When setting the batch size, consider the number of APNs generated by your business processes and the amount of memory available on your system.

SYNTAX

```
data_sync [-u=user-name {-p=password | -pf=password-file} -g=group]
{-class=class-name [-filename=file-name | -itemKeyFile=file-name] |
-item_id=template | -key=keyAttr1=keyVal1
[,keyAttr2=keyVal2...,keyAttrN=keyValN]} [-OnlyVIS]
{-site=site-name -sync | -republish | -verify}
[-f=sync | republish | verify] [-status] [-commit_ixr]
[-cleanup_transaction [-transaction_id=transaction-id |
-before_last_process_date=date ] | list_transactions] [-pull] [-update]
[-replacement_site=site-replacing-extinct-site] [-stubs_only] [-sync_file_stubs]
[-force]
[-report[=file-name] [-format[=CSV] [-separator["|" | ","]]]]
[-exclude_files] [-disable_modified_only]
[-exclude=relation-type1 -exclude=relation-type2...]
[-exclude_security_update]
[-include=relation-type1 -include=relation-type2...]
[-include_bom] [-classoffile=class-name] [-revision-selector]
[-latest_ds_version] [-assert_extinct_ods] [-assert_extinct_site]
[-exclude_folder_contents] [-since=YYYY-MM-DD:HH:NN]
[-qry_name=query-name -qry_attr=attr-name1 -qry_val=attr-value1 [-qry_attr=attr-name2
-qry_val=attr-value2...]]
[-batch_size=number-of-objects-per-batch]
[-deferred_batch_size=batch-size-for-deferred-objects]
[-batch_objects=list-of-deferred-classes]
[-batch_file=file-name-listing-deferred-classes]
[-nonbulk]
[-verbose] [-log]
[-checkpoint [-compress_ind_files={S | I | N} ] ]
[-transaction_id] [-restart]
[-optionset | -optionset=optionset-name [-de_incl_rlz_bom]
[-workset_include_relz_de] [-4gd_id=object-id -class=4gd-class-name] ]
[-include_4gd_baseline_content]
[-override_options=option1:value1,option2:value2, ...]
[-session_options=option1:value1,option2:value2, ...]
```

[-bp]
[-h]

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID. This is generally a user with administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

Caution:

For HTTP enabled sites, remote site operations log on using the default group for the user supplied with the **-u** argument. Any value supplied with the **-g** argument is ignored.

-pull

Specifies synchronization starts in **pull** mode, that is, from a replica site.

For 4GD objects, you must specify the **-optionset** argument. This causes the utility to pull the data as a TC XML payload.

-class

Specifies the class of objects to be searched to determine what objects need synchronization. This does not mean that all objects of the given class are synchronized; only those that were modified since the last time they were exported to the given site(s) are synchronized. See restrictions 1 and 2.

-filename (fn)

Specifies the name of the input file containing IDs or names of objects to update. See restriction 1.

-itemKeyFile

Specifies the name of the input file containing the list of item key strings of the items you want to update. The following listing shows sample content of a file for updating a list of items:

```
item_id=export_001
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

If the item key file has 4GD object key strings, the corresponding 4GD class must be supplied using the **-classoffile(cof)=** argument. The following listing shows sample content of a file for updating 4GD data:

```
4gd_id=DE_Export_001
4gd_id=DE00001_ID,object_name=DE00001_Name,object_desc=DE00001_Desc
```

The **4g_id** entry maps to the corresponding unique ID of the 4GD class, for example:

```
Class=Cpd0DesignElement, 4gd_id=cpd0_design_element_id
```

-item_id (item)

Specifies the ID or template of items to update. See restriction 1.

-key

Specifies the item keys of the items to update, the template of the item keys, or the 4GD object key. It is mutually exclusive with the **-item_id** argument. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

-OnlyVIS

Specifies synchronization of only visualization datasets attached directly or indirectly to a replicated item revision with status. See restriction 8.

If you use the **-OnlyVIS** argument, the **-batch_size** value defaults to 500.

-f

Specifies the function to be performed; define one of the following functions:

sync

Initiates the update process. See restrictions 2 and 4.

republish (repub)

Republishes objects that have been modified since last published. See restriction 2.

verify (veri)

Note:

Siemens Digital Industries Software recommends that you always use the **-item_id=*** argument with the **verify** function. If you use the **-class=item** argument with the **verify** function, it processes only the items that have been modified after their last export.

When used with the **-update** argument, deletes the IXRs of objects for which replicas do not exist at the remote sites.

When used without the **-update** argument, generates a report. See restriction 2.

The report returns the following verification verdict codes:

0	Object does not exist.
1	Object exists as a primary copy.
2	Object exists as a replica.
3	Object was replaced by a POM stub.

status (stat) Displays the status of a given transaction ID.

If the **-site** argument is given, only status of the given sites is displayed. If no **-site** argument is given, the status returned depends on whether the command is given at the site that initiated the checkpointed transaction or the site is the receiving end of the transaction. If the command is given at the initiating site and no **-site** argument is given, the status of the local site and all target sites is returned.

commit_ixr (cmi) Updates the export records at the owning site once the data is known to have been successfully imported at a target site.

Under normal conditions, the update of the export records are performed automatically by each subprocess that succeeds in completing the send operation to its assigned site. Use this function only if either of the following conditions occur:

- The failure occurs at the importing site, and the user performs the restart using the **item_import** utility.

- The failure occurs after the data is successfully imported by a target site, but a failure occurs just before or during the updating of the export records.

You must use at least one **-site** argument to identify the site or sites for which export records are to be updated.

You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have **read** access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an error message is displayed to **stdout** and the utility returns a nonzero value.

cleanup_transaction (ct) Removes transient data generated during a checkpoint transaction or fast sync transactions if used with the **-optionset** argument.

For checkpoint transactions, this transient data consists of the export data and supporting directories and files used to manage the transaction. You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have **delete** access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an error message is displayed to **stdout** and the utility returns a nonzero value.

For fast sync transactions, you must specify either the **-transaction_id** or **-before_last_process_date** argument.

-before_last_process_date (blpd) Specifies the date used to determine which fast sync transactions to clean up. The date must be supplied in the following format:

YYYY-MM-DD:HH:NN:SS

YYYY represents the four-digit year value. MM represents the two-digit month value. DD represents the two-digit day of the month. HH represents a two-digit hour value from 0 to 23. NN represents a two-digit minute value from 0 to 59 and SS represents a two-digit second value from 0 to 59. The DD:HH:NN:SS are optional. If they are not specified, the utility uses 12:00 AM of the specified date.

Valid only with the **-cleanup_transaction** argument.

list_transactions (lt)

Lists all uncleaned checkpoint transactions. An uncleaned transaction is one in which its transient data has not been deleted from the transfer area using the **cleanup_transaction** function.

- Active transactions can only be detected at the site that initiated it. The receiving end of a transaction is not able to tell if a transaction is active or not.
- The list of inactive transactions initiated by the local site and the list of transactions initiated by remote sites are based only on the contents of the transfer area of the node where this command is executed.
- The list of active transactions initiated by the local site is always complete because it is based on data stored in the local database.

You must execute this function at a node or host that has direct access to the transfer area where the export was performed (if at initiating site) or where the data was transmitted to by the owning site (if at receiving site). You must also have *read* access to the operating system directory where the export data is placed within the transfer area. If these conditions are not met, an appropriate error message is displayed to **stdout** and the utility returns a nonzero value.

-assert_extinct_ods (aeo)

Deletes all publication audit record (PAR) objects from the local database for an ODS that no longer exists. This removes any record about objects previously published to the ODS and makes it possible to delete the published objects at a later time. Only the **-site** and **-login** switches are required. This is valid only with the **-f=verify** argument. See restriction 5.

-assert_extinct_site (aes)

Deletes all import export record (IXR) objects from the local database for a site that no longer exists. This removes any record of objects previously exported to the site and makes it possible to delete the exported objects at a later time. Only the **-site** and **-login** arguments are required. This is valid only with the **-f=verify** argument. See restriction 5.

-replacement_site (rs)

Specifies the name of the site that replaces the site to be extinct. This argument is valid only with the **-assert_extinct_site** argument. All objects owned by the extinct site are redirected to the replacement site.

-stubs_only

Specifies that stubs are processed only when the **-replacement_site** argument is specified. This argument is valid only with the **-assert_extinct_site** argument.

-sync_file_stubs

Processes dataset files excluded from export and updates remote stubs if needed. This argument is valid only when the **-class=ImanFile** argument is specified.

-update (upd)

Performs a database update. Must be given in order for the **-f=sync**, **-f=republish**, or **-f=verify** to occur; otherwise, it only does a dry run and generates a report. See restriction 4.

-report

Generates a synchronization report. If a file name is not supplied, the report is displayed in a shell. When published to a file, the default format is as a comma-delimited .csv file. To use a pipe (|) as a separator, include **-format** and **-separator = "|"**. For example,

```
data_sync -u=tcadmin -p=password -item_id=iteamA012 -sync -optionset
-update -site=site2 -report=report.csv -format=CSV -separator="|"
```

-site

Specifies the Teamcenter site to update. This argument can be used multiple times in the command line to synchronize with multiple name-identified sites.

-exclude_files (exf)

Excludes dataset files. See restriction 6.

-exclude

Excludes the specified relation type. This argument may be given multiple times and must use the database name (not the display name) of the relation type. See restriction 6.6.

For best TC XML performance, use a closure rule for relation instead of include/exclude.

-exclude_security_update

Excludes items with only changed project or license security data. When these items are not required to be synchronized, excluding them can significantly improve synchronization performance. Items with other changes in addition to project or license security data changes will be synchronized.

-include

Includes the specified relation type. This argument may be given multiple times and must use the database name (not the display name) of the relation type. Use this argument to force the inclusion of a relation type that may have been excluded during the last export.

For best TC XML performance, use a closure rule for relation instead of include/exclude.

-exclude_folder_contents (efc)

Excludes the contents of a folder. Intended for use with NX part families where family members are stored in a folder that is related to the item.

-include_bom (bom)

Synchronizes all components of an assembly. This synchronization includes any newly added components to the existing assembly. See restriction 6.

-disable_modified_only (dmo)

Disables the default behavior of synchronizing subobjects inside an item only if they were modified since the last time the item was exported.

Normally, this argument is not used. See restriction 6.

-revision-selector

Valid only if both the **-f=sync** and **-update** arguments are specified. Choose one of the following revision selectors:

-all_revisions

Synchronizes all revisions.

-latest_revision

Synchronizes only the latest revision, regardless of the release status. This is the default if no revision selector is specified and more than one site is to be synchronized. If synchronizing only one site, the default selector is **same_as_last_export**.

-latest_working

Synchronizes only the latest working (unreleased) revision.

-latest_released

Synchronizes only the latest released revision with any release status.

-latest_working_or_any

Synchronizes the latest working revision; if no working revision, synchronizes the latest released revision of any release status.

-release_status = release-status-type

Synchronizes only the latest released revision with the specified release status type.

-all_released_revs

Synchronizes all revisions with a release status including in-process item revisions.

-same_as_last_export

Synchronizes using the options used the last time the item was exported. This is the default if no revision selector is specified and only one site is being synchronized. If synchronizing multiple sites, the default selector is **latest_revision**.

-include_pfmembers

Identifies the related part family members to be exported when handling part family templates.

-include_pftemplates

Identifies the related part family template to be exported when handling part family members.

-pf_bom_treatment

Identifies the part family objects associated with the assemblies to be exported. The argument must be used in conjunction with the **-include_bom** argument. Valid arguments are:

-members

Includes part family member components present in the assembly.

-templates

Includes part family template rather than part family member components.

-all

Includes both the part family member components and templates.

-none

Includes neither the part family member components nor the templates.

-latest_ds_version (ldv)

Synchronizes only the latest version of datasets. See restriction 6.

-force

Synchronizes objects regardless of whether they were modified since the last time they were exported. See restriction 3.

-since

Synchronizes only those objects modified since the specified date and time, which must be specified in YYYY-MM-DD:HH:NN format, where YYYY is the year; MM is the month number from 1 to 12; DD is the day from 1 to 31; HH is the hour from 0 to 23, and NN is the minute from 0 to 59. HH and NN are optional and default to zero, which indicates 12 a.m. of the given date. This is valid only with the **-class** argument.

-verbose

Displays maximum amount of information when the utility is run in verbose mode. Typically, nonverbose utility sessions only display error messages. Do not abbreviate this argument to **-v**.

-log

Places detailed information in the **data_sync.log** file. The information includes the start and ending time for each step performed by the **data_sync** utility. Use this argument to analyze the performance of the utility.

-checkpoint (cp)

Initiates a checkpoint transaction, that is, a transaction that can be restarted at the point of failing. This argument is valid only when both **-f=sync** and **-update** are specified. If specified without the **-update** argument, this argument is ignored.

Valid only with **-f=sync**.

If a noncheckpoint operation is initiated for multiple target sites and some target sites are not currently available based on a preliminary availability check, Teamcenter sends a message to **stdout**

to notify the user about unavailable sites, removes unavailable sites from the target site list, and then performs the operation for the available sites.

compress_ind_files (cif)

Specifies compression mode to use to compress files in the export directory during a checkpoint transaction. If not specified, creates a single large ZIP file. This argument is valid only with the **-checkpoint** argument. Valid values are:

- **S**

Creates a single large ZIP file.

- **I**

Creates a ZIP file for each individual file, resulting in multiple ZIP files.

- **N**

No files are compressed.

-transaction_id (trid)

Specifies a 14-character transaction ID for a given checkpoint-related operation or fast sync transaction.

-optionset

Sends the data using the TC XML functionality. This method provides better performance for large data transfers and must be used when exporting 4GD data.

You can replicate 4GD objects to a remote site when you specify the **-optionset** argument with the **send** function. Optionally specify the transfer option set used for the export by setting **-optionset** to the set name. If you do not specify an option set, the utility uses **MultiSiteOptSet** as the default transfer option set value. Values of the options listed in the option set govern the object export. The option set must exist at the exporting site.

For 4GD objects, you can also specify the **-de_incl_rlz_bom** and **-workset_incl_relz_de** arguments.

-de_incl_rlz_bom

Sends the source objects of a design element (Type:**Cpd0DesignElement**). You must specify the **-optionset** argument and specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument.

-workset_incl_relz_de

Sends the source objects of a design element (**Cpd0DesignElement**) in a Workset (**Cpd0Workset**). You must specify the **-optionset** argument and specify a 4GD object using the **-4gd_id**, **-key**, or **-itemKeyFile** argument.

-4gd_id

Specifies a 4GD object identifier or 4GD object pattern. The **-class=class-name** argument must be specified with the **-4gd_id** argument.

The utility maps the **-4gd_id** argument to the corresponding unique ID of the 4GD class, for example:

```
Class=Cpd0DesignElement, 4gd_id=cpd0design_element_id
```

A 4GD partition object and 4GD subset definition objects do not have a unique 4GD class ID. Therefore, using **-4gd_id** for partition objects or subset definition objects may result in the update of multiple objects

To export unique partition object use multifield key attributes supplied in the **-key** argument, see [Examples](#).

-restart (rs)

Restarts a given transaction at the point of failure.

Valid only with the **-f=send** function.

-qry-name

Runs a saved Query Builder that defines the list of objects to process. One or more **-qry_attr** and **-qry_val** pairs specify the attribute value pairs the objects must match to be processed.

-batch_objects (bo)

Specifies a list of comma-separated deferred classes. The list must not contain spaces.

-batch_file (bof)

Specifies the file name of a text file containing a list of deferred classes. Each class name is contained on a separate line.

-batch_size (bs)

Specifies the number of objects to synchronize per batch. A new process is created for each batch. All workspace objects (not just items) that are synchronized are considered part of a batch. The default batch size is 2000. The maximum value you can specify is 99999. If you enter a value greater than 99999, the utility sets the value of **-batch_size** to the default.

To process items one item at a time, use the **-non-bulk** argument.

-deferred_batch_size (dbs)

Specifies the number of objects per batch; a new process is created per batch. The default value is 2000. This value must be a positive integer. Use this argument to process thousands of objects to avoid memory and disk shortage problems.

The following classes are supported for deferred objects:

- **Dataset**

- Folder
- Form
- ImanRelation
- MEAppearancePathNode
- NamedVariantExpression
- PSOccurrence
- VariantExpression
- VariantExpressionBlock

-nonbulk

Process one object at a time when synchronizing a list of objects. By default, **data_sync** runs with **-batch_size** set to a value of **2000**. Overriding this default operation with **-nonbulk** may result in slower performance, but can aid debugging when researching batch synchronization issues.

-override_options

Override the following default operations by setting the following values to **true**. (The default value of each is **false**.) Options and values are separated by colons. Multiple overrides are separated by commas.

opt_exclude_apn	When syncing BOM structures, set to true to exclude APNs from being synchronized.
opt_include_rendering	Set to true to include the rendering dataset in the synced data.
opt_include_ice	When syncing EngChange lists, set to true to sync IncrementalChangeElement items.
opt_exp_all_wso	Set to true to sync secondary objects for all relations. Use with care as this may result in a very large amount of data being synchronized.

-session_options

Override the following default session operations by setting the following values to **true**. (The default value of each is **false**.) Options and values are separated by colons. Multiple overrides are separated by commas.

forceUpdateOnImport	Set to true to force an update during a sync if a replica's last saved date (lsd) value or last modified date (lmd) value is greater than the primary object's value.
opt_traverse_by_island	Set to true to sync only an island of data. Do not sync across islands.

-bp

Displays best practices information.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

1. One of the following arguments must be supplied: **-class**, **-filename**, or **-item_id**.
2. One of the following arguments must be supplied: **-f=sync**, **-republish**, or **-f=verify**.
3. The **-force** argument can only be used along with the **-filename** or the **-item_id** argument. It does not function when used in combination with the **-f=verify** argument.
4. Unless the **-update** argument is given, the **data_sync** utility generates only reports.
5. The **-assert_extinct_site** and **-assert_extinct_ods** options can only be used with the **-f=verify** argument.
6. The **-exclude_files**, **-exclude=**, **-include_bom**, **-disable_modified_only** and **-latest_ds_version** options can be used if both the **-f=sync** and **-update** arguments are supplied.
7. The **-classoffile** argument currently supports only the **Item**, **ItemRevision**, **Dataset**, **Form**, **Folder**, **Role**, **User**, **Group**, and **Person** classes.
8. The **-include** and **-update** arguments must be supplied with the **-OnlyVIS** switch.
9. To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
10. For 4GD data, traversal-free synchronization is not supported in pull mode.

All arguments supported for push mode with the **-optionset** argument are supported in pull mode.

EXAMPLES

Required log-in information is omitted from the following examples.

- To generate a report of items that must be synchronized for a given site:

```
data_sync -class=Item -site=Site1 -f=sync
```

The report is output to **stdout**. No synchronization is performed.

- To synchronize all items copied to a site and output a report to a file:

```
data_sync -class=Item -site=Site1 -f=sync -update -report=report.lst
```

(The default revision selector, **-same_as_last_export**, is used.)

- To synchronize the latest released revisions of items:

```
data_sync -class=Item -site=Site1 -f=sync -update -latest_released
```

- To synchronize all forms and datasets:

```
data_sync -class=Form -class=Dataset -site=Site1 -f=sync -update
```

- To republish all previously published items to the **Mfg_ODS** ODS:

```
data_sync -class=Item -site=Mfg_ODS, -f=republish -update
```

- To check if datasets copied to the **Design_Center** site still exist and delete the IXR from the primary if a copy is no longer there:

```
data_sync -class=Dataset -site=Design_Center -f=verify -update
```

- To force synchronization of a list of items specified in a text file copied to a site and output the report to a file:

```
data_sync -filename="/myhome/itemlist.txt" -classoffile=Item  
-site=Site1 -f=sync -update -force -report=report.lst
```

- To force synchronization of a single item or items that match a template:

```
data_sync -item_id=Eng* -site=Site1 -f=sync -update -force  
-report=rep.lst
```

- To synchronize an out-of-date 4GD design element and output the report to a file:

```
data_sync -4gd_id=Ste2_DE0486 -class=Cpd0DesignElement
-site=Site1 -optionset -sync -update
-report=4GD_update_rep.txt
```

- To destroy all the export records to a known extinct site:

```
data_sync -u=Tc-admin-user -p=password -g=group -site=XSite -f=verify
-update -assert_extinct_site
```

- To destroy all the publication records to a known extinct ODS site:

```
data_sync -u=Tc-admin-user -p=password -g=group -site=XSite -f=verify
-assert_extinct_ods
```

- To destroy export records, BVRs, and attachments of specific deleted replica item revisions:

```
data_sync -site=S1 -f=verify -update -fn=mylist
-cof=ItemRevision
```

The **mylist** file has item revision names in the following format: **item123/A**

- To destroy export records of specific deleted replica datasets:

```
data_sync -site=S1 -f=verify -update -filename=mylist
-classoffile=Dataset
```

The **mylist** file has dataset names in the following format: **dataset123**

- To start synchronization in **pull** mode:

```
data_sync -pull -class=Item -site=S1 -update
-report=report.lst
```

- To force synchronize a list of items specified in a text file in **pull** mode:

```
data_sync -pull -filename="/myhome/itemlist.txt" -force
-update -report=report.lst
```

- To generate a report of which items must be synchronized in **pull** mode:

```
data_sync -pull -filename="/myhome/itemlist.txt" -site=S1
-f=sync -report=report.lst
```

- Include the rendering dataset and **IncrementalChangeElement** items in the synchronized data:

```
data_sync -sync -update -site=site2 -item_id=xxxx
          -override_options=opt_incl_rendering:true,opt_include_ice:true
```

- Force updating of the data on the import site even though the last saved dates are the same for the primary and replica:

```
data_sync -sync -update -site=site2 -item_id=xxxx
          -session_options=ForceUpdateOnImport:true
```

- To synchronize visualization datasets that are under a replicated item revision that has status:

```
data_sync -OnlyVIS -since=2005-01-01:01:01 -site=Site1 -f=sync
          -update -report=report.lst
          -include=IMAN_Rendering -include=IMAN_specification
```

- To check if items copied to the **Design_Center** site still exist and delete the IXR from the primary if a copy is no longer there, enter the following command on a single line:

```
data_sync -item_id=* -site=Design_Center -f=verify -update
```

- To delete the IXRs of objects whose replicas do not exist at the remote sites, enter the following command on a single line:

```
data_sync -item_id=* -site=Site1 -f=verify -update
          -report=rep.lst
```

- To generate a report of the IXRs, enter the following command on a single line:

```
data_sync -item_id=* -site=Site1 -f=verify -report=rep.lst
```

Both this and the previous example generate reports listing all objects including those that are no longer at the remote site.

- To synchronize all items copied to a site and output a report to a file with newly added components to existing assembly:

```
data_sync -class=Item -site=Site1 -f=sync -update
          -include_bom -report=report.rpt
```

- To synchronize any particular item transferred to a replica site and output a report to a file with newly added components to existing assembly:

```
data_sync -u=Tc-admin-user -p=password -g=group -item_id=Item1
          -site=Site1 -f=sync -update -include_bom -report=report.rpt
```

- To synchronize all **imanfile** objects copied to a site and output a report to a file:

```
data_sync -class=imanfile -site=Site1 -f=sync -update
-report=report.lst
```

- To force synchronization of **imanfile** objects for all datasets specified in a text file copied to a site and output the report to a file:

```
data_sync -filename=/myhome/datasetlist-for-imanfiles.txt
-classoffile=imanfile -site=Site1 -f=sync -update -report=report.lst
```

- To synchronize files and initiate a checkpoint for three sites:

```
data_sync -f=sync -checkpoint -item_id=item123
-site=Site2 -site=Site3 -site=Site4 -update
```

- To force synchronization of files and initiate a checkpoint for three sites:

```
data_sync -f=sync -update -checkpoint -item_id=item123
-site=Site2 -site=Site3 -site=Site4
```

- To check the status of a given transaction:

```
data_sync -f=status -transaction_id=AhEZaOnRAAMfD
```

- To restart a given transaction for a given site:

```
data_sync -f=sync -transaction_id=AhEZaOnRAAMfD
-restart -site=Site3
```

- To synchronize all 4th Generation Design (4GD) objects copied to a site and output report to a file:

```
data_sync -class=Cpd0DesignElement -site=Site1
-sync -update -report=report.lst
```

- To synchronize specific 4GD objects copied to a site and output report to a file:

```
data_sync -4gd_id=DE000001 -class=Cpd0DesignElement
-site=Site1 -sync -update -report=report.lst
```

- To clean up fast sync transactions prior to specific last process date, list the available transactions to get the last process dates:

```
data_sync -optionset -lt
```

Clean up the transactions:

```
data_sync -optionset -ct -blpd=2012-12-18:20:10:00
```

- To synchronize **MEAppearancePathNode** and **VariantExpression** objects in batch mode with **deferred_batch_size**:

```
data_sync -item_id=<Top Item>
-bo=MEAppearancePathNode,VariantExpression -bs=3000 -update -sync
-include_bom -site=site2 -dbs=3000
```

- To use the **TC_cms_relation_optset_map** preference to include or exclude relations:
 1. Add the relation and option set to the **TC_cms_relation_optset_map** preference, for example:

```
IMAN_rendering, opt_rel_rendering
```

2. In the PLM XML/TC XML Export Import Administration application, expand the **TransferOptionSet**, click **MultiSiteOptSet**, and add the **opt_rel_rendering** option with the default value set to **false**.
3. Expand **ClosureRule**, click **MultiSiteDefaultCR**, and add the following clause:

```
CLASS:WorkspaceObject:CLASS:Dataset:RelationP2S:IMAN_rendering:
SKIP:$opt_rel_rendering==false;
```

This clause states, from any **WorkspaceObject**, find the dataset using **IMAN_rendering** relation, and when the relation is **opt_rel_rendering**, skip the dataset during export. It means the default is to always exclude the **IMAN_rendering** relation for an exported object.

4. To include the **IMAN_rendering** in the synchronization using the **data_sync** utility, type:

```
data_sync -include=IMAN_rendering -optionset
```

IMPORTANT NOTES

1. When synchronizing items, all item revisions, BOM view revisions, BOM views, forms, and datasets associated with the item will also be synchronized. However, in some cases the item itself is not modified, so the last modification date is not updated and, therefore, cannot be used as the sole basis for synchronization. In most cases, it is necessary to specify all classes associated with an item to guarantee that complete synchronization is accomplished. This means that the command to run the **data_sync** utility should include several class switches, for example:

```
data_sync -class=Item -class=ItemRevision -class=PSBOMViewRevision
```

Note:

If your database contains a large number of replicated items (more than 10,000), you should synchronize one class at a time. When doing so, you should begin with the **Item** class, and

then the **ItemRevision** class, followed by the **PSBOMViewRevision** class, and continue down the schema to dataset and forms classes.

2. The **PSBOMViewRevision** class must be specified instead of the **PSBOMView** class so that changes to the structure is synchronized.
3. When synchronizing an assembly, the **data_sync** utility does not automatically traverse the assembly tree. Rather, it synchronizes each subassembly or component individually on an as-needed basis. If you want the utility to traverse the assembly tree, use the **-include_bom** argument.
4. When synchronizing an assembly, **data_sync** transfers new components that are part of the assembly, when sending an assembly with the **-include_bom** argument set to true.
5. Because the **data_sync** utility never involves any transfer of ownership, there is no need to perform export recovery if the utility terminates prematurely.
6. When synchronizing, the utility performs an automatic verification. It checks if the object being synchronized still exists at the remote site prior to synchronizing it. If a replica no longer exists, the utility deletes the corresponding IXR.
7. The **-verbose** argument can be used to analyze the performance of the **data_sync** utility. The **-verbose** argument prints the system times at important stages during the process of synchronization.
8. Siemens Digital Industries Software recommends that you synchronize only one site at a time rather than synchronizing multiple sites in a single run of the **data_sync** utility. This allows you to use the **-same_as_last_export** revision selector that uses the same import/export options used to replicate the item. If you must synchronize multiple sites, create a script that loops through sites but only invokes the **data_sync** utility with only one site at a time.

USING FOLDERS WITH THE DATA_SYNC UTILITY

Folders can be used with the **data_sync** utility, as shown below:

```
data_sync -filename=/tmp/folderlist -classoffile=Folder...
```

If the content of the folder has changed since the last export, if references have been added or removed, the **data_sync** utility updates the remote copy to reflect the current state of the folder.

If no references have been added or removed from a folder since the last export, it is not considered to have been modified. Therefore, if the objects referenced in the folder have changed and need to be updated at the remote site using the **-classoffile=Folder** argument, use the **-force** argument.

GENERATING REPORTS

This example shows how to generate a report called **data_sync.rpt** against the Detroit site:

Enter the following command on a single line:

```
data_sync -class=Item -verify -report=data_sync.rpt -site="Detroit"
```

The results are as follows:

```
Object Date Last Modified Site Date Last Exported Type (Class)
-----
DS_0401_02A 1997-04-03 15:13:50 Detroit 1997-04-03 12:47:45 Text (Dataset)
DS_0401_02A;1 1997-04-03 15:13:40 Detroit 1997-04-03 12:47:48 Text (Dataset)
DS_0401_02A;2 1997-04-03 15:13:43 Detroit 1997-04-03 12:47:52 Text (Dataset)
0320_01/A 1997-03-24 15:34:44 Detroit 1997-03-24 15:33:57 Text (Dataset)
0320_01/A;1 1997-03-24 15:34:33 Detroit 1997-03-24 15:34:00 Text (Dataset)
0320_01/A;2 1997-03-24 15:34:38 Detroit 1997-03-24 15:34:04 Text (Dataset)
0320_01/A;3 1997-03-24 15:34:42 Detroit 1997-03-24 15:34:06 Text (Dataset)
sueD0324-4;1 1997-03-24 22:04:44 Detroit 1997-03-24 21:57:28 Text (Dataset)
sueD0324-4;2 1997-03-24 22:04:48 Detroit 1997-03-24 21:57:32 Text (Dataset)
```

ERROR CODES

Error code 100228 indicates that a Multi-Site Collaboration file transfer operation has failed. The most likely causes are a network connection failure or an abort (crash) of the IDSM process at the remote site. For the former, retry the operation. For the latter, examine the IDSM system log files at the remote site.

diff_xml

Uses the **bomwriter**-generated output files with the **grdvua_on** option specified at two days, compares the PLM XML files, and generates a difference XML file. This difference XML file contains all of the changes performed on the assembly structure. This XML file is then used to update the audit log file dataset.

SYNTAX

```
diff_xml -u= user-name {-p=password | -pf=password-file} [-g=group] -item=item-id -rev=revision-id
-key=key-id [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the item ID of the top node of the assembly structure.

-rev

Specifies the revision ID of the top node of the assembly structure.

-key

Specifies the key of the object. The **-key** argument can be used instead of the **-item** argument.

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Open the Teamcenter menu shell with the database connection variables set and then execute the following command:

```
diff_xml -u=Tc-admin-user -p=password -g=group -item=000125 -rev=001
```

distributed_execute

Executes the **item_report** utility, both locally and remotely, and generates reports.

You can specify any command line parameters required for the **item_report** utility on the **distributed_execute** command line, and those arguments are passed to the **item_report** utility.

Note:

This utility does not support individual item ID input. You must use the **-itemidsfile=file** argument.

Siemens Digital Industries Software recommends you first perform the **-distributed_func=traverse_items** step. This accumulates all traversed items from all specified sites and collects them in an output file (BOM traversal). You can use this file as input argument in subsequent steps, for example, report.

This utility does not collect logs at remote sites and return them to the local machine.

Siemens Digital Industries Software also recommends you test this utility with emphasis on:

- Verifying the utility performs the same way locally and remotely.
- Receiving required report files and test miscellaneous combinations of command line parameters; any additional parameters specified are passed to the calling program.

SYNTAX

```
distributed_execute [-u=user-id {-p=password | -pf=password-file} -g=group]
-distributed_func=function -itemidsfile=datafile
-distributed_sites=site1,site2,site3 -outfile=file-name
[-delimiter=delimiter-character] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter

database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-distributed_func

Specifies one of the following functions:

traverse_items	Traverse BOM on all specified sites and produce union of all traversed IDs, argument maps to the item_report utility argument -traverseditemfile .
report	Generates a distributed IDSM-based report based on list of items input from an ID file, argument maps to item_report utility, generate the reports, and merger reports.

-itemidsfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item IDs. This argument is required.

-outfile

Specifies the output file. This argument is required.

-distributed_sites

Specifies a list of sites, both local and remote, on which this command is executed. This argument is required.

-delimiter

Specifies a delimiter character for the output file. The default value is the vertical bar (|). Ensure the delimiter in the site-based file and merge file input match.

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- To execute **item_report** to generate a list of traversed objects for three sites (user, password, and group arguments are not shown in the example):

```
distributed_execute -distributed_sites=Site1,Site2,Site3  
                  -outfile=trav.out -distributed_func=traverse_items  
                  -itemidsfile=item_id.txt
```

- To execute **item_report** to generate report files and merge file (user, password, and group arguments are not shown in the example):

```
distributed_execute -distributed_sites=Site1,Site2,Site3  
                  -outfile=merge.out -distributed_func=report  
                  -itemidsfile=item_id.txt
```

dsa_util

Distributes system administration data, such as users and groups, from one site to another. When adding a new site, this allows you to enter the site information of all sites in the network so the new site can exchange data with them.

Caution:

Do not use this utility to share organization objects between sites if the same organization objects are shared through a global organization.

Note:

This utility should be used only for the initial migration of system objects. Siemens Digital Industries Software recommends that you do *not* use this utility to maintain system objects.

Propagates Teamcenter administration data among multiple sites and allows administrators to:

- Manage system data from a central site.
- Support non-networked sites.
- Create reports of the results of a distribution operation.

SYNTAX

```
dsa_util [-u=user-id {-p=password | -pf=password-file} -g=group]
-f={list_sites | check_sites |
set_logging_level -level={FATAL | ERROR | WARNING | INFO | DEBUG |
TRACE | OFF}}
[-site=remote-site1-name -site=remote-site2-name...]
[-h={topics | topic-name} | -h [-f=function-name | -class=class-name]]
```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

Caution:

For HTTP enabled sites, remote site operations log on using the default group for the user supplied with the **-u** argument. Any value supplied with the **-g** argument is ignored.

-f

Identifies the function to perform. Must be one of the following functions:

list_sites (ls)	Lists all the sites that are defined in the local database.
check_sites (cs)	Checks the availability of all sites defined in the local database. A site is considered available for Distributed System Administration purposes if its IDSM server is ready.
set_logging_level (sll)	Sets the logging level at the remote site indicated by the -site argument to the level indicated by the -level argument. See restrictions 5 and 6 .

-level

Specifies the logging level for a given remote site in a remote procedure call (RPC)-based Multi-Site environment. Valid values are:

- **FATAL**

Logs only severe errors that cause premature program termination.

- **ERROR**

Logs other run-time errors or unexpected conditions.

- **WARNING**

Logs run-time situations that are undesirable or unexpected, such as use of deprecated APIs.

- **INFO**

Logs informational messages that highlight the progress of the application at a coarse-grained level, such as startup and shutdown events.

- **DEBUG**

Logs detailed information on the flow through the system. This information is useful for debugging an application.

- **TRACE**

Logs the most detailed information on system events and operation.

- **OFF**

Turns off logging.

See restrictions [5](#) and [6](#).

Set the logging level at the local site in the **logger.properties** file in the *TC_DATA* directory.

-site

Identifies the remote sites to which system objects are distributed. May be given multiple times in the same command line to distribute to multiple sites.

-h

Displays help information on basic usage.

RETURN VALUES

Return value 0
upon success

Return value >1
upon failure

RESTRICTIONS

1. To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
2. Do not use this utility to share organization objects between sites if the same organization objects are shared through a global organization.

3. When exporting the user object, this utility does not export the license level of the user. The license level of the user is set to the lowest available license level at the importing site. System administrators at the importing site must manually set the license level and/or the license bundle of each user.
4. The **dsa_util** utility does not recognize externally managed users, groups, roles, persons, and group members with a **datasource** attribute value greater than **0** and convert them to remotely managed (for example, managed by an LDAP external directory at a remote site). Because the **dsa_util** utility is the only way user constructs can be converted to remotely managed, no user construct objects in the Organization user interface appear as remotely managed.
5. The **set_logging_level** function and **-level** argument are only for RPC-based Multi-Site environments. Use the JMX console to set the logging levels for the loggers in an HTTP (four-tier) Multi-Site environment.
6. To use this argument, you must add the source site (site where the utility is run) to the **IDSM_dsa_sites_permitted_to_push_admin_data** preference value at the remote site.

EXAMPLES

To set the Multi-Site logging level to **DEBUG** at a remote site **203456177**, enter the following command on a single line:

```
dsa_util -u=Tc_admin-user -p=password -f= set_logging_level
        -site=203456177 level=DEBUG
```

DumpCMSConfigInfo

Searches in the org.eclipse.ui.activities section of Teamcenter *plugin.xml* extension files to find commands, views, and perspectives and their Teamcenter IDs, and dumps them to a set of .csv files. This utility is part of the **com.teamcenter.rac.util** plug-in and is run as a separate application. The utility must be run from the *TC_ROOT\portal* directory.

SYNTAX

```
teamcenter -application com.teamcenter.rac.util.DumpCMSConfigInfo
-u=user-ID {-p=password | -pf=password-file} [-g=group]
[-mode=activity | popup | view | appContext | All]
[-nosplash]
[-nl=en_US | de_DE | ... ]
[-path=c:\temp]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

Note:

If your Teamcenter server uses Security Services single sign-on, see *Before you begin* for additional information.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-mode

Specifies the options. This utility can dump perspective ID, popup command ID, view ID, and application context ID information into CSV format. This parameter is optional; the default is **all**.

Valid parameters are:

activitypopupviewappContextall

Parameter	Information type	Output file name
activity	perspective	activityxxx.csv
popup	command	commandsxxx.csv
view	view	viewsxxx.csv
appContext	application context	appContextxxx.csv

-nosplash

Suppresses the application splash screen.

-nl

Locale specified by the administrator in which to see the view, perspective, and command names. The default is taken from the OS locale.

-path

The directory location to output .csv files. If not specified, then the current directory is used.

-h

Displays help information on basic usage.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

SAMPLE USAGE

Windows

```
TC_ROOT\rac\Teamcenter.exe -application
com.teamcenter.rac.util.DumpCMSConfigInfo -nosplash -nl en mode=all
-path=D:\test
```

Linux

```
TC_ROOT/rac/Teamcenter -application
com.teamcenter.rac.util.DumpCMSConfigInfo -nosplash -nl en mode=all
-path=/u/test/
```

email_polling

Performs email polling using a specified rule. The email_polling utility is useful for testing polling configuration, or can be called by a cron job as an alternative to using Teamcenter Dispatcher.

PREREQUISITES

- Email polling type is configured.
- Email polling rule is created.
- Email polling settings are configured.

SYNTAX

```
email_polling [-u=user-id {-p=password | -pf=password-file} -g=group-name]
[-rule_id=email poll rule object ID] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-rule_id

Specifies the email poll rule object id.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Manually configure the Teamcenter environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To poll an email folder using the **CustomApplicationRule1**:

```
email_polling -u=myusername -p=mypassword -rule_id=
CustomApplicationRule1
```

ensure_site_consistency

Allows users to perform corrective actions if the site ownership transaction is interrupted due to a system or network crash or a user-initiated process termination (such as the Windows Task Manager). In cases where legitimate error conditions are encountered (such as lack of transfer privilege or duplicate item IDs), there is no requirement to perform any corrective action; Teamcenter restores the data to consistent states under most non-crash conditions.

Note:

This utility should typically be run only at the exporting site; not at the importing site. The flag that marks an object as requiring this utility is always at the exporting site.

When working with TC XML Multi-Site, use **-f=offline_recovery** in place of **-f=recovery**. When using **-f=offline_recovery**, the **ensure_site_consistency** can be used at both the export and the import site.

SYNTAX

```
ensure_site_consistency [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=clean_all_rec | list_all_rec | offline_recovery | recovery | report | set_owning_site
{ [-item_id=item-id ] | [-key= {keyAttr1=keyVal1 [,keyAttr2=keyVal2],...[,keyAttrN=keyValN]
[-class=class-name] } | [-folder=folder-name] | [-filename=file-name] |
[-itemKeyFile=file-name] | [-search] | [-4gd_id=object-id -class=4gd-class-name] }
[-report=file-name]
[-mode={sst | gms | full | min | auto} ]
[-remote_site=site-name]
[-real_owning_site=site-name]
[-optionset=option-set-name]
[-override_options={option1:value1 [,option2:value2],...[,optionN:valueN]}
[-session_options={option1:value1 [,option2:value2],...[,optionN:valueN]}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the function to perform. Valid values are:

clean_all_rec Deletes all inconsistent local/replica **IXR**, **ITXR**, and **PAR** records specified in the input that exist for local/replica objects at a local site. This argument deletes workspace objects only. It does not delete **VariantExpression**, **AbsOccData**, or **MEApprPathNode** objects.

Inconsistent local/replica **IXR**, **ITXR**, and **PAR** records occur in the following situations:

- The object is local and the referencing auxiliary objects (**IXR**, **ITXR**, and **PAR**) are replicas.
- The object is replica and the referencing auxiliary objects (**IXR**, **ITXR**, and **PAR**) are replicas.
- The object is replica and the referencing auxiliary objects (**IXR**, **ITXR**, and **PAR**) are replicas.
- At a hub site, the object is local/replica and the referencing auxiliary objects (**IXR**, **ITXR**, and **PAR**) are replicas.

list_all_rec Lists all inconsistent local/replica **IXR**, **ITXR**, and **PAR** records that exist for local/replica objects on a local site. This value must be used with the **report** value.

offline_recovery Performs offline ownership recovery operations. When working with TC XML Multi-Site, use **-f=offline_recovery** in place of **-f=recovery**.

If the item has an SST dataset, use **-f=recovery** to recover the ownership.

recovery Perform recovery operations such as reclaiming site ownership, releasing transfer locks, and removing unwanted export records.

report Generates a list of objects that require recovery. The list is output to a text file identified by the value of the **report** argument. By default, the report contains the Teamcenter Integration Framework transfers (GMS) report followed by the Synchronous Site Transfer (SST) report. If the **-mode** argument is supplied with this argument, the utility generates a report on the mode specified.

The **clean_all_rec** value can be used with this value.

set_owning_site Sets the owning site of the input objects to the site specified by **-real_owning_site**.

-folder

Specifies a folder that contains items on which to perform corrective action.

The use of a folder is intended for Workspace objects that do not have unique IDs, for example, datasets and forms. This is useful for failed remote checkins of multiple objects where many of the remotely checked-out objects do not have unique IDs, for example, datasets, forms, BVRs, and so forth.

-filename

Specifies a file name that contains a list of items on which to perform corrective action. The file should only contain item IDs. This argument is mutually exclusive with the **-folder**, **-item_id**, and **-search** arguments.

-class (cl)

Specifies the Teamcenter class of the object specified by the **-name** or **-4gd_id** argument. This argument is valid only with the **-name** or, when the **-low_level** argument is specified, with the **-4gd_id** argument. The default class is **Item**.

For organization objects, this argument accepts **Role**, **User**, **Group**, and **Person** classes.

-key

Specifies the keys of the objects on which to perform corrective action. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

-itemKeyFile

Specifies the name of the file containing the keys of the objects on which to perform corrective action.

-search

Finds all the objects that are flagged as requiring corrective action.

This argument is mutually exclusive with the **-folder**, **-filename**, and **-item_id** arguments.

- When used with the **report** function, the utility generates a report on all the objects that are found as requiring corrective actions.
- When used with the **recovery** function, the utility performs corrective actions on the objects that are found as requiring corrective actions.

-4gd_id

Specifies a 4GD object identifier or 4GD object pattern. The **-class=class-name** argument must be specified with the **-4gd_id** argument.

The utility maps the **-4gd_id** argument to the corresponding unique ID of the 4GD class, for example:

```
Class=Cpd0DesignElement, 4gd_id=cpd0design_element_id
```

A 4GD partition object and 4GD subset definition objects do not have a unique 4GD class ID. Therefore, using **-4gd_id** for partition objects or subset definition objects may result corrective action on multiple objects.

To export unique partition objects, use multifield key attributes supplied in the **-key** argument. See *Examples*.

-item_id

Specifies the item ID.

- When used with the **report** function, the utility generates a report on the item specified by *item-ID*.
- When used with the **recovery** function, the utility performs corrective action on the item specified by *item-ID* only if the specified item is flagged as requiring corrective actions.

-report

Specifies the output file path for generating the report. Use this argument with either the **report** function or the **recovery** function.

- When used with the **report** function, the report lists the objects that require corrective action.
- When used with the **recovery** function, the report lists the objects where corrective action was taken.

-mode

Specifies the recovery method type of transfer failures that you want to recover or the type of report to generate when used with the **-report** argument. If you do not specify this argument, the utility uses standard Multi-Site Collaboration transfer failures or generates both a Synchronous Site Transfer (SST) and Teamcenter Integration Framework (GMS) report when used with the **-report** argument. You can specify one of the following valid values:

- **sst**

When used with the **recovery** function, **sst** recovers SST transaction failures.

When used with the **report** argument, the report lists the SST transaction objects that require corrective action.

- **gms**

When used with the **recovery** function, **gms** recovers GMS transaction failures.

When used with the **report** argument, the report lists the GMS transaction objects that require corrective action.

- **full**

When used with the **offline_recovery** function, **full** reimports objects from the metafile and restores site ownership to the local site.

- **min**

When used with the **offline_recovery** function, **min** reads the object UUIDs from the metafile and flips their ownership. A full reimport is not performed. **min** is valid only if the metafile was generated with a transfer of ownership.

- **auto**

When used with the **offline_recovery** function, **auto** restores ownership of the specified item. Objects are not reimported.

-remote_site

Use to convert the replica item to be locally owned (changing the replica to the primary). When used with the **offline_recovery** function and with **-mode=auto**, **-remote_site** specifies the last site for which transfer of ownership was attempted. All objects owned by this site within the given item are owned by the local site. **-remote_site** is not valid with **-real_owning_site**.

-real_owning_site

When used with **f=offline_recovery** with **-mode=auto**, ownership is assigned to the item on the target site specified by **-real_owning_site**. The target site's ISDM must verify that the item exists on the site and that the item on the target site is the item master.

When used with **f=set_owning_site**, ownership is assigned to the item on the site specified by **-real_owning_site** with no ISDM verification.

-optionset

Specifies the name of a transfer option set to use when updating exported objects. The option set must be available at exporting site. The values of the options in the option set are applied to the exported objects.

-override_options

Used with **-optionset** to override options defined in the named option set. Specify options to override with comma-separated name-value pairs. Delimit multiple name-value pairs with semicolons.

-session_options

Used with **-optionset** to override session options defined in the named option set. Specify options to override with comma-separated name-value pairs. Delimit multiple name-value pairs with semicolons.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Generate a report on all the objects that are flagged as requiring corrective actions:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=report -search -report=recovery_candidates.txt
```

- Generate a report on the item specified by *item_ID*:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=report -item_id=000301 -report=recovery_item.txt
```

- Perform corrective actions on all the objects that are flagged as requiring corrective actions:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=recovery -search -report=recovery_fixup.txt
```

- Perform corrective actions on the item specified by *item_ID*:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=recovery -item_id=000301 -report=recovery_fixup.txt
```

- Perform corrective actions on a list of item IDs:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=recovery -filename=item_id_list.txt
```

The **item_id_list.txt** file should contain a list of item IDs, one item ID per line.

- Perform corrective actions on all objects under a given uniquely named folder:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=recovery -folder=RecoveryFolderFor26June2007
```

- To restore ownership on an item whose id is **MyCorruptItem**:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=offline_recovery -mode=auto -item_id=MyCorruptItem
-remote_site=Site2
```

- To restore ownership on objects contained in a metafile without reimporting:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=offline_recovery -mode=min -dir=c:\metafile_dir
```

- To reimport objects from the metafile and restore site ownership:

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=offline_recovery -mode=full -dir=c:\metafile_dir
```

- Set the item and all of its dependent objects to be owned by Site2. Collect the dependent objects using the **MultiSiteExpOptSet** option set.

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=set_owning_site -item_id=item001 -real_owning_site=Site2
-optionset=MultiSiteExpOptSet
```

- Set the item and all of its dependent objects to be owned by Site2. Collect the dependent objects using the **MultiSiteExpOptSet** option set and set the option **opt_incl_rendering** with the value **true**.

```
ensure_site_consistency -u=Tc-admin-user -p=password -g=group
-f=set_owning_site -item_id=item001 -real_owning_site=Site2
-optionset=MultiSiteExpOptSet -override_options=opt_incl_rendering:true
```

export_recovery

Recovers and restores exported objects to your database under certain conditions. Occasionally, when you export an object and transfer ownership the object may not be successfully imported at the destination site. This places the object in an undefined state where no one has ownership. The preferred method of correcting this situation is to have the destination site complete the import/export transaction by importing the object into the database from the importing site's **TC_transfer_area** (using interactive object import).

However, if this is not possible, the **export_recovery** utility is used to restore the object to the exporting database from the exporting site's **TC_transfer_area** using the **min** or **full** mode (effectively canceling the export/transfer ownership transaction). If no data is available at either site, recovery can be attempted by running the automode at the exporting site that was the last known owning site.

Use the **export_recovery** utility when an export with transfer of site ownership fails, resulting in objects within an item having inconsistent site ownership. The mode of recovery to use depends on whether there is a valid export directory. The directory must include the **objects.meta** file.

Siemens Digital Industries Software recommends the following order for attempting export recovery procedures; you should try the succeeding procedure only if you cannot perform the previous one or if the previous one fails to restore site ownership:

- If a valid export directory exists (most likely in the **TC_transfer_area** of the exporting or importing site), use either **full** or **min** mode while specifying the valid export directory with the **-dir=** switch. If you attempt to recover at the exporting site, use **min** mode; if you attempt to recover at the importing site, use **full** mode.
- If a valid export directory does not exist, you must attempt recovery from a valid database copy that may be a replica or one with inconsistent site ownership. Use **export_recovery** in **auto** mode. Specify the **-include_bom** switch if appropriate. Specify **-exclude** and/or **-include** switches, if desired.
- If the **auto** mode fails to restore site ownership, perform the manual export recovery procedure:
 1. Define the **TC_EXPORT_COPY=TRUE** environment variable.
 2. Run **item_export** as a Teamcenter administrative user to transfer site ownership to any site.
 3. Run **export_recovery** in **min** mode specifying the directory output in step 2 as the **-dir=** parameter.
 4. If successful, delete the export directory from step 2.

SYNTAX

```
export_recovery [-u=user-id {-p=password | -pf=password-file} -g=group]
               -mode={ full | min | auto | find }
               [-item_id=item-id-to-restore]
```

```
| [-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
| [-folder=folder-name] | [-filename=file-name]
| [-itemKeyFile=file-name] | [-dir=directory]
[-report=report-file] [-remote_site=last-transfer-site] [-include_bom]
[-real_owning_site=desired-owning-site]
[-exclude=relation-type1 -exclude=relation-type2 ...]
[-include=relation-type3 -include=relation-type4 ...]
[-ignore_am_rules] [-update_lmd] [-bp] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies the basic mode in which the utility operates. The value of this argument can be one of the following:

full

Restores objects from the export metafile, imports them in to your database and restores ownership to your site.

min

Restores ownership to your site without reimporting data from the metafile. Valid only if the metafile was generated with transfer of ownership.

auto

Restores ownership on the specified item without reimporting. You must specify the **-itemid** argument and either the **-remote_site** or **-real_owning_site** arguments when using this mode.

find

Searches for items with inconsistent site ownership and generates a report.

-dir

Defines the path of the directory containing the exported metafile and the data files. Required only with the **-mode=full** and **-mode=min** arguments.

-item_id

Specifies the ID of the item to process. Wildcards are allowed.

-folder

Defines the name of the Teamcenter folder containing the list of items to process.

-filename

Defines the full path of the file that contains the list of items to process.

-key

Specifies the keys of the items to process. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

-itemKeyFile

Specifies the name of the input file containing the keys to process. The file format is:

```
- key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...
- key =[keyAttr1=keyVal1][keyAttr2=keyVal2]...
- key =[keyAttr1=keyVal1][keyAttr2='keyVal2']...
```

-remote_site

Defines the last site for which a transfer of ownership was attempted. This argument is valid only with the **auto** mode.

-report

Specifies the full path of the report file. Valid only with **find** mode.

-real_owning_site

Changes the owning site of specified objects to the site designated. Valid only with the **-mode=auto** argument.

-include_bom

Includes assembly components, if any exist.

-exclude

Excludes the specified relation type and may be given multiple times. The database name (not display name) of the relation type must be used.

-include

Includes the specified relation type and may be given multiple times. The database name (not the display name) of the relation type must be used. Use this argument to force the inclusion of a relation type that is not specified by your **TC_relation_required_on_export** preference.

-ignore_am_rules

Ignores AM rules for recovery purposes.

-update_lmd

Updates the last modified user and date. Valid only with the **-mode=auto** argument.

-bp

Displays best practices information.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

- To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
- At least one primary mode must be specified.
- Not more than one primary mode can be specified.
- For the **-mode=auto** and **-mode=find** options, exactly one object selection filter (**-itemid**, **-filename**, or **-folder**) must be specified.

EXAMPLES

In each of the following examples, the **-u=user-id** **-p=password** and **-g=group** arguments are assumed:

- To restore ownership on an item with the ID **MyCorruptItem**:

```
export_recovery -mode=auto -item_id=MyCorruptItem
               -remote_site=Manufacturing
```

- To restore ownership on objects contained in an export metafile without reimporting:

```
export_recovery -mode=min -dir=metafile_dir
```

- To reimport objects from the metafile and restore site ownership:

```
export_recovery -mode=full -dir=metafile_dir
```

- To generate a report of ownership inconsistencies:

```
export_recovery -mode=find -filename=suspect_itemlist.dat
               -report=report.dat
```

- To make an item (**xyz**) in the local site a replica that is owned by another site (**Site2**):

```
export_recovery -mode=auto -item_id=xyz -real_owning_site=Site2
```

- To restore ownership of an entire assembly:

```
export_recovery -mode=auto -item_id=Assy1 -remote_site=Site2
               -include_bom
```

generate_admin_data_compare_report

Generates an interactive report of differences between administration data in two Teamcenter environments. To make the comparison, the utility requires an administration data export package from one environment as a source, and either the local environment or another administration data export package as a target. If data is the same in both the source and the target, the report shows that there are no differences.

If an object is referenced by other objects, the report includes a where-used table that indicates the categories and objects that have references to the current object in both source and target environments.

The report summary page shows all the administration data types included in the comparison and the number of differences for each element present within the category. The report includes a glossary of administration data categories and the elements available in each of the categories.

SYNTAX

```
generate_admin_data_compare_report -u=user-ID {-p=password | -pf=password-file} -g=group
-sourcePackage=path-to-source-package
-targetPackage=path-to-target-package | -extractAndCompare
-adminDataTypes={Admin-data1,Admin-data2,...,Admin-dataX | all}
-outputDir=path-to-directory-for-report-files
[-listTypes]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-sourcePackage

Specifies the full path, including the file name, of the source administration data package.

-targetPackage

Specifies the full path, including the file name, of the target administration data package. This argument is mutually exclusive with the **-extractAndCompare** argument.

-extractAndCompare

Extracts the administration data from the current environment and compares it with source package identified by the **-sourcePackage** argument. This argument is mutually exclusive with the **-targetPackage** argument.

-adminDataTypes

Specifies the types of administration data to include in the report. You provide the data types as a comma-separated list (no spaces) or you can specify **all** to get all supported data types in the report.

Tip:

Use the **-listTypes** and **-sourcePackage** arguments to get a list of available administration data types.

If the report contains multiple data types, it includes a where-used table showing where each object is referenced.

-outputDir

Specifies the path to directory where you want the compare report saved. You must specify this argument.

-listTypes

Displays a list of the administration data types at the source that you can compare to the target. You must specify the **-sourcePackage** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

This is a Java utility that, by default, has the maximum Java heap size set to 1024M. For reports that contain a large number of objects, you may need to increase maximum Java heap size to avoid

out-of-memory errors or poor performance. If possible, set the maximum heap to at least to 4096M for large reports. You can set this value using the **BMIDE_SCRIPT_ARGS** environment variable, for example:

```
set BMIDE_SCRIPTS_ARGS=-Xmx4096M
```

Note:

Java standards require that no more than 25 percent of total RAM be allocated to virtual memory (VM). If the amount allocated to the Java VM exceeds this percentage, degradation of performance can occur.

FILES

As specified in Log files produced by Teamcenter.

EXAMPLES

- Display a list of the administration data types that you can compare to the source environment package:

```
generate_admin_data_compare_report -u=admin-username -p=admin-password
-g=dba
-sourcePackage=C:\temp\admin_data\SiteA\siteA.zip -listTypes
```

- Generate a comparison report of the preferences at the source and target:

```
generate_admin_data_compare_report -u=admin-username -p=admin-password
-g=dba
-sourcePackage=C:\temp\admin_data\siteA\siteA.zip
-targetPackage=C:\temp\admin_data\siteB\siteB.zip
-adminDataTypes=Preferences
-outputDir=C:\temp\admin_data\compare_sites_A_and_B_preferences
```

- Generate a comparison report of the organization data at the source and target:

```
generate_admin_data_compare_report
-u=admin-username -p=admin-password -g=dba
-sourcePackage=C:\siteA\siteA.zip -extractAndCompare
-adminDataTypes=Organization
-outputDir=C:\temp\admin_data\compare_organization_between_local_and_A
```

generate_admin_data_report

Generates a report showing the specified administration data for the site where you run the utility or for an export package. The export package can be from a remote site.

The report contains HTML pages for the administration data objects, showing their properties with hyperlinks to referenced objects. If an object is referenced by other objects, its HTML page contains a where-used table that indicates the categories and objects that have references to the current object.

The report has a summary showing all the administration data types included in the report and the instances of each element present within the category. The report also has a glossary page with descriptions of the administration data categories and the elements available in each of the categories.

SYNTAX

```
generate_admin_data_report -u=user-ID {-p=password | -pf=password-file}
-g=group
-adminDataTypes=Admin-data1,Admin-data2,...,Admin-dataX | all
[-inputPackage=input-package-path]
-outputDir=path-to-directory-for-report-files
[-listTypes]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Teamcenter Utilities* in Teamcenter help.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-adminDataTypes

Specifies the types of administrative data to include in the compare report. You provide the data types as a comma-separated list (no spaces). You may also specify the **all** value to include all data types defined in the local system or the specified input package.

Tip:

Use the **-listTypes** argument to get a list of available administration data types.

If the report contains multiple data types, it includes a where used table showing where each object is referenced.

-inputPackage

Specifies the full path, including the file name, of the export administration data package from the site for which the report is generated. If you do not specify this argument, the utility generates a report for the local site.

-outputDir=

Specifies the path to directory where you want the report saved. You must specify this argument.

-listTypes

Displays a list of the available administration data types that you can include in the report.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Manually configure the Teamcenter environment* in *Teamcenter Utilities* in Teamcenter help.

This is a Java utility that, by default, has the maximum Java heap size set to 1024M. For reports that contain a large number of objects, you may need to increase maximum Java heap size to avoid out-of-memory errors or poor performance. If possible, set the maximum heap to at least to 4096M for large reports. You can set this value using the **BMIDE_SCRIPT_ARGS** environment variable, for example:

```
set BMIDE_SCRIPT_ARGS=-Xmx4096M
```

Note:

Java standards require that no more than 25 percent of total RAM be allocated to virtual memory (VM). If the amount allocated to the Java VM exceeds this percentage, degradation of performance can occur.

FILES

EXAMPLES

- Generate a list of the administration data types that you can export:

```
generate_admin_data_report -u=admin-username -p=admin-password -g=dba  
-listTypes
```

- Generate a report containing the preferences and their values at the local site:

```
generate_admin_data_report -u=admin-username -p=admin-password -g=dba  
-adminDataTypes=Preferences  
-outputDir=C:\temp\admin_data\siteA\preferences_report
```

- Generate a report containing the Access Manager and Organization administration data from an export package of a remote site:

```
generate_admin_data_report -u=admin-username -p=admin-password -g=dba  
-adminDataTypes=AccessManager,Organization  
-inputPackage=C:\temp\admin_data\siteB\siteB.zip  
-outputDir=C:\temp\admin_data\siteB\am_and_organization_report
```

idsminetd

Serves as the Integrated Distributed Services Manager (IDSM) launching program on Linux systems. Located in the **\$TC_ROOT/bin** directory, it is run at system startup and services all inbound requests for a new IDSM.

For more information on IDSM, see *Multi-Site Collaboration*.

SYNTAX

```
idsminetd [-u=user-id {-p=tcp-port-number | -pf=password-file} -g=group]
[-d] [-t] [-r=idsm-start-script]
[-n=RPC-program-number]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the port number on which the IDSM should run. Default is the system-assigned port number.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-d

Specifies debug mode for stand-alone testing. The server runs in the foreground.

-t

Enhances logging.

-r

Specifies the IDSM start script.

-n

Specifies the RPC program number the IDSM should use. The default RPC program number is used if this argument is omitted.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

The selected port number must be in the range 1025–65535 and must not conflict with other system services.

EXAMPLES

Under normal circumstances, this utility runs only at system startup. The following is an example of running a debug session:

```
idsminetd -d -t -p=33333 -r=/tmp/myscript
```

import_file

Imports files into the Teamcenter database according to a set of user-specified arguments. These arguments supply user identification information, dataset information, and (optionally) item information to be associated with the imported file. The arguments may be specified on the command line to import a single data file or in a file to import multiple data files (bulk import).

Depending on the arguments, each data file is copied (an **ImanFile** object is created), a dataset is created (or modified), and if specified, an item is created or modified to contain the dataset. In the absence of a specified item, the dataset is placed in the user's **Newstuff** folder.

The **import_file** utility does not support the creation of custom item types.

SYNTAX

```
import_file [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=file-name | -i=file-name [-vb] [-log=file-name] -type=datasettype -d=dataset-name
-ref=named-reference [-de={n | e | a | r}] [-item=item-id | -itemkey=key-id]
[-itemRevUid=item-revision-uid]
[-relationType=relation-type] [-use_ds_attached_to_rev_only]
[-revision=item-rev-num] [-ie={n | y}] [-desc=string]
[-v=volume-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Imports a single file into Teamcenter. The full path must be provided if the file does not reside in the current working directory. The **-f** and **-i** arguments are mutually exclusive. See example 1.

-i

Imports multiple files into Teamcenter using a specified import file. The full path must be provided if the file does not reside in the current working directory. The **-f** and **-i** arguments are mutually exclusive. See example 2.

-vb

Runs utility in verbose mode. Displays maximum amount of information. Nonverbose sessions only display error messages.

-log

Creates a log of items and datasets created.

-type

Defines the dataset type in Teamcenter, for example, **Text** or **UGPART** datasets. **-type** is case sensitive and must match the dataset type name as defined in the Teamcenter data model.

-d

Specifies the name of the dataset into which the file is imported.

-ref

Specifies the type of named reference associated with the file. The value specified by this argument may or may not be identical to the value specified by the **-type** argument.

For example, **Text** or **UGPART** type datasets have named references of **Text** and **UGPART**, respectively. However, **DirectModel** type datasets have a **JTPART** named reference.

Each dataset type defines one or more named references to be associated with it. See restriction numbers 1 and 2.

-de

Indicates that a dataset exists. Used when a dataset of the same name already exists.

=n

Specifies that a new dataset be added even if one with the same name exists. If it does exist, it is added to the same item folder. If it does not exist, it is placed in the new item folder or the user's **Newstuff** folder.

=e

Specifies that a new dataset be added if the dataset name specified by the **-d** argument is not used by any existing dataset. The utility displays an error message if you supply this argument and the dataset name specified by the **-d** argument already exists in Teamcenter.

=a

Specifies that the imported file be added as a named reference to the existing dataset. When this is done, a new dataset version is created. This version contains the additional imported named reference file.

=r

Specifies that a new dataset revision be created and the existing named reference be replaced with the new one. This option generates an error if the dataset has no existing named reference.

-item

Specifies the name of the item containing the dataset that references the imported file.

-itemkey

Specifies the key of the object. You can use the **-item** argument or the **-itemkey** argument.

To find the key of an object, use the **get_key_string** utility.

-itemRevUid

Specifies the 14 character UID string of the item revision object where the file and dataset are to be attached.

-relationType

Specifies a relation to use when the **IMAN_specification** relation is not appropriate. If the **-relationType** argument is not specified, the **IMAN_specification** relation is used.

-use_ds_attached_to_rev_only

Specifies a dataset based on its name and type, but it considers datasets attached to the specified item and item revision only. The item and item revision are specified by the **-item** and **-revision** parameters, respectively.

This prevents the utility from referring to datasets with the same name and type, but that are unattached or are attached to another item or revision.

If you want to revise the existing dataset instead of creating a new one, the **-use_ds_attached_to_rev_only** parameter is particularly useful when used along with the **de=r** argument.

-revision

Specifies the item revision number and revision ID. See restriction number 3.

-ie

Specifies behavior if the item already exists.

=n

Specifies that the dataset will not be added if the item already exists.

=y

Specifies that the dataset may be added if the item already exists. If the item exists, but the item revision does not, an item revision is created.

-desc

Specifies a user-defined text description of an item that is created by the import function. If the **import_file** utility is creating a new revision of an existing item, this is the description of the item revision.

-v

Specifies the full path of the Teamcenter volume where the imported file is placed.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. To create a dataset in Teamcenter, the user must specify the dataset type and the named reference.
2. When importing a file as a dataset, you must specify the named reference using the **-ref** argument.
3. When importing a file into an item revision, you must specify the revision; otherwise, an error message displays indicating a missing revision.

EXAMPLES

- To import a single operating system file, **bike.dat**, into Teamcenter as a **UGPART** dataset named **my_bike_dataset**, enter the following on a single line:

```
$TC_ROOT/bin/import_file -user=user-id -p=password -g=group -f=bike.dat
-type=UGPART -d=my_bike_dataset -ref=UGPART
```

- To import multiple operating system files into Teamcenter, first create an input file that contains the following information:

```
-f=bike1.dat -d=my_bike1_dataset -type=UGPART -ref=UGPART
-f=bike2.dat -d=my_bike2_dataset -type=UGPART -ref=UGPART
```

```
.
-f=binkeN.dat -d=my_bikeN_dataset -type=UGPART -ref=UGPART
```

- Run the **import_file** utility using the input file from example 2, entering the following command on a single line:

```
$TC_ROOT/bin/import_file -user=user-id -password=password -group=group
-i=input-file
```

- Import the **d:\some_file.jt** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\some_file.jt -type=DirectModel -d=my_jt_file_dataset
-ref=JTPART
```

- Import the **d:\WordDoc.doc** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\WordDoc.doc -type=MSWord -d=my_word_dataset -ref=word
```

- Import the **d:\ExcelFileTest.xls** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\ExcelFileTest.xls -type=MSEExcel -d=my_excel_dataset
-ref=excel
```

- Import the **d:\myfile.txt** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\myfile.txt -type=Text -d=my_text_file_dataset -ref=Text
```

item_export

Exports a single item or multiple items in batch mode. It is the companion to the **item_import** utility. This utility supports part family templates and members and works with the **TC_relation_required_on_export** and **TC_relation_required_on_transfer** preferences.

SYNTAX

```
item_export [-u=user-id {-p=password | -pf=password-file} -g=group]
-dir=directory
{-item=item-id | -keykey-id [-rev=revision-selector] | -filename=input-file
| -itemKeyFile=file-name}
{-owning_site=site-name | -target_site=site-name1, site-name2, ...}
[-exclude=relation-type1 -exclude=relation-type2...]
[-include=relation-type1 -include=relation-type2...]
[-reason=export-reason] [-latest_ds_version] [-include_bom]
[-batch_objects=list-of-deferred-classes]
[-batch_file=file-name-listing-deferred-classes]
[-deferred_batch_size=batch-size-for-deferred-objects]
[-preview] [-report=file-name] [-continue_on_error]
[-xfr_top_lvl_only] [-xfr_top_asm_only] [-xcl_files]
[-status=release-status] [-exclude_folder_contents]
[-classoffile=class-name] [-separator=separator-character]
[-dont_exclude_protected] [-email=email-address] [-script=script-name]
[revision-selector] [-include_bc] [-include_supercedures] [-v] [-h]
```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-dir

Specifies the full path of the directory where the metafile and data files are stored.

-item

Specifies the ID of the item to be exported. Valid only if no input file is specified using the **-i** argument.

-key

Specifies the key of the object. You can use the **-item** argument or the **-key** argument.

To find the key of an object, use the **get_key_string** utility.

-rev

Identifies the revision to be exported. This can be the revision ID or one of the following keywords:

```
LATEST
LATEST_WORKING
LATEST_RELEASED
LATEST_WORKING_OR_RELEASED
USE_STATUS
```

If the **USE_STATUS** keyword is given, you must specify a release status using the **-status** argument. This is valid only when you are not transferring site ownership.

If used with an input file (**-i** argument), the revision keyword is used for every item in the input file. Keywords cannot be specified using the command line; however, you can use revision selectors at the command line as discussed below.

Note:

The revision ID cannot be specified when using an input file.

-filename (fn)

Specifies the name of an input file that contains the list of item IDs to be exported.

The format of the text file must contain the **-item=** prefix to each item ID. For example, to export item IDs **002259**, **002260**, and **002261**, the input file contains the following entries:

```
-item=002259
-item=002260
-item=002261
```

-keyFileName

Specifies the name of the input file containing the keys to be exported. The file format is:

```
-key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...
-key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...
-key = [keyAttr1=keyVal1][keyAttr2='keyVal2']...
```

-classoffile (cof)

Specifies the class of objects contained in the input file. If no class is specified, the default class is **Item**. Valid only with the **-filename** argument.

-separator (sep)

Specifies the character to separate the item and revision IDs in the file. The default is **/**.

-target_site (ts)

Specifies the export target site or sites. If more than one site is specified, sites must be separated by a comma and the entire string must be enclosed in quotes. Either the **-target_site** or **-owning_site** argument is required.

-owning_site (os)

Specifies the site to which ownership is transferred. Either the **-target_site** or **-owning_site** argument is required.

-exclude (exc)

Specifies the relation type to be excluded. This argument may be given multiple times, and the database name (not the display name) of the relation type must be specified. You cannot exclude the **IMAN_master_form** and **TC_ic_intent_rtype** relation types with or without ownership transfer. Also, you cannot exclude the **IMAN_RES_audit** with ownership transfer.

-include (inc)

Specifies the relation type to be included. This argument may be given multiple times, and the database name (not the display name) of the relation type must be specified. You cannot include the **IMAN_RES_checkout** and **IMAN_based_on** relation types.

-exclude_folder_contents (efc)

Excludes the contents of a folder. Intended for use with NX Part families where family members are stored in a folder that is attached to the item.

-dont_exclude_protected (dxc)

Does not exclude export-protected objects. If set, any export-protected object within an item prevents the export of the entire item.

-reason (rea)

Specifies the reason for exporting to sites. Up to 240 characters.

-latest_ds_version (ldv)

Exports only the latest version of datasets; default is to export all versions. Valid only when site ownership is not being transferred.

-include_bom (bom)

Exports all components if the given item is an assembly.

-preview (pre)

Performs an export dry run and generates a report to the file specified by the **-report** argument. If the **-report** argument is not specified, the report is output to the screen.

-report (rep)

Outputs preview or completion reports to the specified file. If no report file name is specified, the report is output to the screen.

-continue_on_error (con)

Continues the export operation even if errors are detected on optional objects. Optional objects are attachments other than requirement, specification, or primary form objects.

-xfr_top_lvl_only

Only transfers ownership on top-level items specified in the input file.

-xfr_top_asm_only

Transfers ownership only on the top-level assembly items, as specified in the input.

-xcl_files

Excludes export of files in datasets.

-include_pfmembers

Identifies the related part family members to be exported when handling part family templates.

-include_pftemplates

Identifies the related part family template to be exported when handling part family members.

-pf_bom_treatment

Identifies the part family objects associated with the assemblies to be exported. The argument must be used in conjunction with the **-include_bom** argument. Valid options are:

-members

Includes part family member components present in the assembly.

-templates

Includes part family template rather than part family member components.

-all

Includes both the part family member components and templates.

-none

Includes neither the part family member components nor the templates.

-status (stat)

Specifies the release status type to use for selecting the item revision to be exported.

-include_bc

Exports the change revision along with the **BOMChange** objects associated with the affected assemblies of the change revision.

-include_supercedures

Exports the change revision along with the supercedures associated with the **BOMChange** objects.

-email

Specifies the email address to which the export report is sent.

The default address is stored in the Teamcenter user account.

-script

Specifies the name of the script in the **TC_BIN** directory that is executed after a successful export. If a script is already defined by the **TC_post_export_script** preference, the specified script overrides the preference entry.

-revision_selector

Determines which item revision is exported with the item. The valid selectors are as follows:

latest_revision (lt)	Exports the latest revision only, regardless of whether it is a working or released revision.
latest_working (lw)	Exports the latest working revision only.
latest_released (lr)	Exports the latest released revision only with any release status.
latest_working_or_any (lwoa)	Exports the latest working revision. If no working revision exists, it exports the latest released revision with any release status.
status (stat)	Specifies the release status to be exported.

If no revision selector is given, all revisions are exported.

Note:

Revision selectors should be capitalized only when used with the **-rev=** switch and should be in lower case when used as a switch.

-v

Runs utility in verbose mode to display maximum amount of information. Typically, nonverbose utility sessions only display error messages.

-batch_objects (bo)

Specifies a list of comma-separated deferred classes. If you use this argument with the **preview** argument, only nondeferred objects with the number of deferred objects appear in the report.

-batch_file (bof)

Specifies the file name of a text file containing a list of deferred classes.

-deferred_batch_size (dbs)

Specifies the number of objects per batch; a new process is created per batch. The default value is **1000**. This value must be a positive integer. Use this argument to process thousands of objects to avoid memory and disk shortage problems.

The following classes are supported for deferred objects:

- **Dataset**
- **Folder**
- **Form**
- **ImanRelation**
- **MEAppearancePathNode**
- **NamedVariantExpression**
- **PSOccurrence**
- **VariantExpression**
- **VariantExpressionBlock**

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

1. The **-item** argument is mutually exclusive with the **-i** and **-filename** arguments.
2. Either the **-target_site** or **-owning_site** argument must be specified and must not be a local site.
3. It is the responsibility of the user exporting objects to inform the system administrator which directories need to be exported and to which site.
4. It is the responsibility of the system administrator to set up the list of other sites which are known to the local site.
5. It is the responsibility of the system administrator to send directories of the exported objects to the receiving sites, users, volumes, and other systems.
6. Administration object types cannot be exported.

EXAMPLES

To restart a checkpoint transaction that failed during import:

```
item_export -transaction_id=AjEZaOnRAAAFFD -restart
```

item_import

Imports multiple items (in batch mode) into the Teamcenter database. It is the companion to the [item_export](#) utility. This utility supports part family templates and members.

SYNTAX

```
item_import [-u=user-id {-p=password | -pf=password-file} -g=group]
-dir=directory
[-folder=folder-name] [-preview] [-report=file-name] [-filename=file-name]
[-classoffile=class-name] [-list_metafile] [-include_pfmembers=part-family-members]
[-include_pftemplate=part-family-templates]
[-part_family_bom_treatment={members | templates | all | none}]
[-script=pre-import-script] [-email=email-address]
[-parallel=number-of-parallel-processes] [-continue_on_error] [-verbose]
[-transaction_id=tid] [-restart] [-h]
```

ARGUMENTS

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-dir

Specifies the path name to the directory containing the metafile and the data files to be imported.

-folder (fol)

Specifies the destination folder in which imported items are placed. If the folder does not exist or the argument is not supplied, the imported items are placed in the user's **Newstuff** folder.

-preview (pre)

Performs an import dry run and generates a dry run report to the file specified by the **-report** argument. If the **-report** argument is not specified, the report is output to the screen.

-report (rep)

Outputs a preview or completion report to the specified report file. If no report file name is specified, the report is output to the screen.

-filename (fn)

Specifies the name of the text file listing objects for selective import, one name per line.

-classoffile (cof)

Specifies the class of objects contained in the input file. If not specified, the default class is **Item**.

-list_metafile (lm)

Lists only the contents of the metafile; does not import objects.

-include_pfmembers

Identifies the related part family members to be imported when handling part family templates.

-include_pftemplate

Identifies the related part family template to be imported when handling part family members.

-part_family_bom_treatment

Identifies the part family objects associated with the assemblies to be imported. The argument must be used in conjunction with the **-include_bom** argument. Valid options are:

-members

Includes part family member components present in the assembly.

-templates

Includes part family template rather than part family member components.

-all

Includes both the part family member components and templates.

-none

Includes neither the part family member components nor the templates.

-script

Specifies the name of the script to be executed prior to import. If a script is defined by the **TC_post_export_script** preference, this argument overrides the preference setting. If specified as **NONE**, the script defined in the preference file is executed.

-email

Sends email to the user at the specified email address after completion. If no email address is specified, the email address in the user's Teamcenter user profile is used.

-parallel (par)

Specifies the number of processes to be started automatically. If this argument is not specified, the system imports the deferred objects with a sequential process.

-continue_on_error (con)

Specifies that the import operation proceeds when an error has occurred on an optional object, such as a reference or manifestation attachment.

-verbose (v)

Runs the utility in verbose mode to display the maximum amount of information. Typically, nonverbose utility sessions only display error messages.

-transaction_id (trid)

Specifies the transaction ID for a given checkpoint-related operation.

-restart (rs)

Restarts a given transaction at the point of failure.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

EXAMPLES

To restart a transaction at a site where a failure occurred:

```
item_import -transaction_id=transaction-id -restart
```


item_relink

Replaces the external references for a duplicate item and its corresponding replica in a Multi-Site Collaboration environment. The **item_relink** utility works in conjunction with the **item_rename** utility.

The **Bypass** option enables or disables special bypass object protections for Teamcenter administrators, allowing you to freely access any object in the database to perform maintenance. When running this utility, you must use the **Bypass** option, the user ID must be **infodba**, and the OS user account must have read-access to the NX part files to comply with the rules stated in **Bypass UG Part File Verification**.

Caution:

The **item_relink** utility is used only with Multi-Site Collaboration to process production data. Siemens Digital Industries Software recommends that a full backup of your database be performed before running this utility. This allows you to restore the database if the data becomes corrupted.

- **Naming pattern**

Because the **item_relink** utility works in conjunction with the **item_rename** utility, the same naming pattern must be used in both utilities.

- **Bypass NX part file verification**

Each NX part file that is attached to a duplicate is checked against the corresponding NX part file that is attached to the replica. The **item_relink** utility compares the UID strings in the NX part files. The **ug_inspect** utility retrieves the UID strings from the part files. Therefore, it is very important to run the **item_relink** utility using the OS account that has read access to the part files. Usually, the Teamcenter user account, such as **infodba**, is used to run the utility. If UIDs are not the same, the relink process for the duplicate fails. If you are confident about the part files being reconciled, you can use the **-bypass_ugpart** command line argument to bypass this check. The **-bypass_ugpart** argument is ignored if the **item_relink** utility is run in verify mode.

- **Matching criteria**

To find the corresponding replica, construct the replica item ID based on the duplicate item ID and renaming pattern and then search the database for the replica.

To find the corresponding item revision, match the revision ID.

To find the corresponding BOM view, match the view type name.

To find the corresponding BOM view revision, match the view type name.

To find the corresponding secondary object, match the object name, object type, and relation type.

- **Matching results**

For each object that is attached to a duplicate item or duplicate item revision, if more than one object with the same object name, object type, and relation type are found in its corresponding replica item or replica item revision, the first occurrence is used.

If objects attached to a duplicate do not have corresponding objects found in replica, use the **-verify** switch to generate a report that lists any discrepancies. In this case, perform a detailed examination and make the necessary corrections and/or ownership change for the duplicate. If any discrepancies are detected during the relink process, the duplicate is not replaced. Instead, the duplicate is placed in the exception folder. An error message is logged on the report file for review.

- **Exception**

If unexpected Teamcenter internal errors occur or the duplicate contains objects not found in its corresponding replica, the utility stops processing the duplicate that has a problem, logs an error message to the report file, and then processes the next duplicate in the replacement folder. All duplicates that are not reconciled are placed in the **Item_ID_ConsolidationEXP** exception folder so you can further examine these duplicates.

SYNTAX

```
item_relink [-u=user-id {-p=password | -pf=password-file} -g=group]
-replace=folder-name -refile=folder-name
-update | -verify
[-prefix=prefix-removed-from-item-id | -suffix=suffix-removed-from-item-id | -f=file-name]
[-bypass_ugpart ]
[-ignore_attachments]
[-relink_to_latest_rev]
-report | -report=file-name
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-replace

Specifies the name of the folder that holds items that are currently duplicates but should be replicas. This must be the same folder used by the **item_rename** utility.

-verify

Requests verification of compatibility between duplicates and replicas.

-update

Performs the link replacement.

-prefix

Specifies the prefix removed from the item ID of duplicates to form the new item IDs for replicas. This must be the same prefix used by the **item_rename** utility. See restriction 5.

-suffix

Specifies the suffix removed from the item ID of duplicates to form the new item IDs for replicas. This must be the same suffix used by the **item_rename** utility. See restriction 5.

-f

Specifies the file containing item ID cross reference records. The cross reference contains the duplicate item ID and the renamed duplicate item ID for each duplicate item ID. This data is contained in a single 80-byte line in the file. The **item_rename** utility also uses this file. See restriction 5.

-bypass_ugpart

Indicates whether NX part files are verified. If this switch is specified, no verification is performed. This switch is ignored if the **-verify** argument is specified.

-ignore_attachments

Prohibits linking of secondary objects (attachments). Use this argument when the replacement item already has links to all required attachments.

-relink_to_latest_rev

Links all revisions of a duplicate item to latest revision of the replica item. Ignores secondary objects (attachments) to avoid incorrect attachments. This argument cannot be specified for items with multiple views.

-report

Generates a report. Outputs the report to standard output if the file name is not supplied.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

The following restrictions must be understood and adhered to when using the **item_relink** utility:

1. Must be run with the **Bypass** option, and the user ID must be **infodba**.
2. The OS user account must have read-access to the NX part files to comply with the rules stated in **Bypass UG Part File Verification**.
3. The **-replace** and **-refile** arguments must be supplied.
4. Either the **-rename** or **-verify** arguments must be supplied.
5. A default naming pattern is used if the **-prefix**, **-suffix**, or **-f** argument is not supplied.

EXAMPLES

The best practice is to run the **item_relink** utility with the **-verify** argument to do a comparison to find discrepancies between duplicates and replicas. If any exist, examine the discrepancies and make the necessary corrections. To ensure data integrity, Multi-Site Collaboration imposes strict rules on object replication. One of these rules is that only the primary object can be modified. The replicated object must never be checked out for modification or submitted for release. Therefore, if the duplicates contain objects that have no corresponding replicas, the relink process for these duplicates is not performed. However, if the replicated objects have increased with more attachments, the duplicates are overwritten.

- To verify the items in the replacement folder and generate a report called **relink.rpt**, enter the following command on a single line. The naming pattern must be the same as that used by the **item_rename** utility.

```
Item_relink -u=infodba -p=infodba -replace=replacement  
-refile=assm_refile -prefix=AAA -verify -report=relink.rpt
```

- After generating a replacement report, you may need to correct duplicates or change ownership. To replace the links, enter the following command on a single line:

```
Item_relink -u=infodba -p=infodba -replace=replacement  
-refile=assm_refile -prefix=AAA -update -report=relink.rpt
```

item_rename

Changes the item IDs for duplicate part numbers in a naming pattern in a Multi-Site Collaboration environment. The **item_rename** utility works in conjunction with the **item_relink** utility. The main reason for renaming duplicates is to avoid a naming conflict while bringing in copies of the primary data that was previously created.

The **Bypass** option enables or disables special bypass object protections for Teamcenter administrators, allowing you to freely access any object in the database to perform maintenance. When running this utility, you must use the **Bypass** option and the user ID must be **infodba**.

- **Naming pattern**

You can use the **-prefix**, **-suffix**, or **-f** arguments to embed a renaming pattern for the duplicate data objects. If these arguments are not used, the system applies a default naming pattern. The default naming pattern adds the **DUP_** prefix to the duplicate item IDs. For example, if the duplicate item ID is **ABC123**, after the **item_rename** utility runs the duplicate item ID is **DUP_ABC123**.

The **-prefix** and **-suffix** switches enable you to add character strings to the item IDs to form new item IDs.

The **-f** switch supplies a file that contains a list of item ID cross-references. If the **-f** argument is specified, the system ignores the **-prefix** and **-suffix** switches.

- **Cross-reference file format**

The **-f** switch generates a file that contains a list of item ID cross-references, specifically the duplicate item ID and the renamed duplicate item ID. Each set of item IDs is contained in a single 80-byte line. The duplicate item ID precedes the renamed duplicate item ID. The duplicate item ID and the renamed duplicate item ID must be separated by at least one blank space, although more are allowed. Leading blanks may appear before the duplicate item ID or padding blanks may appear after the renamed duplicate item ID.

The system administrator manually creates the cross-reference file. The system administrator must know how to match the item ID replicas and the item ID duplicates.

- **Exception**

If any unexpected Teamcenter internal errors occur, the utility stops processing the duplicate that has a problem, logs an error message to the report file, and then processes the next duplicate in the replacement folder.

The **item_rename** utility is used only with Multi-Site Collaboration.

SYNTAX

item_rename [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group*]
 -replace=*folder-name* -rename | -verify [-prefix= *prefix-added-to-item-id* | -suffix= *suffix-added-to-item-id* | -f=*file-name*] -report=*file-name* [-h]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-replace

Specifies the name of the folder that holds the items that are currently duplicates but that should be replicas.

-prefix

Specifies the prefix added to the item ID of duplicates to form the new item IDs.

-suffix

Specifies the suffix added to the item ID of duplicates to form the new item IDs.

-f

Specifies the file containing item ID cross-reference records. The cross-reference is comprised of the duplicate item ID and the renamed duplicate item ID for each duplicate item ID. This data is contained in a single 80-byte line in the file. The **item_relink** utility also uses this file.

-verify

Requests verification of the existence of renamed items.

-rename

Performs the rename function.

-report

Generates a report and outputs it to standard output if the file name is not supplied.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in the [Log files produced by Teamcenter](#).

RESTRICTIONS

The following restrictions must be understood and adhered to when using the **item_rename** utility:

1. Must use the **Bypass** option, and the user ID must be **infodba**.
2. The **-replace** argument must be supplied.
3. Either the **-rename** or **-verify** argument must be supplied.
4. A default naming pattern is used if either the **-prefix**, **-suffix**, or **-f** arguments are not supplied.

EXAMPLES

The best practice is to run **item_rename** with the **-verify** switch to do a quick search for any objects with the chosen naming pattern. If any exist, choose a different naming pattern for all objects.

- Enter the following command on a single line to verify the items in the replacement folder and generate a report called **rename.rpt**. Assume that the naming pattern adds the prefix **AAA** to the item ID.


```
Item_rename -u=infodba -p=infodba -replace=replacement  
-prefix=AAA -verify -report=rename.rpt
```

If any items in the database have the same item ID as the chosen naming pattern, error messages beginning with *****ERROR** are logged on the **rename.rpt** file.

- Change the naming pattern and run the **item_rename** utility again. Otherwise, use the same command line in step 1, and replace the **-verify** argument with the **-rename** argument to rename the items.

item_report

Generates detail reports of an item or multiple items at the site level. The site level reports can be merged to generate a combined status output.

Using this utility, a site can investigate item consistency and dual ownership. It also provides information about the last modified user, locked information and details about the checkout user (owning and remote checkout). Checkout information includes the checkout user, respective checkout date, and time information.

SYNTAX

```

item_report [-u=user-id {-p=password | -pf=password-file} -g=group]
[-f=report | merge] [-itemidsfile=data-file
| -itemKeyFile=file-name] | -grmtypefile=grm-file
| [-item_id=itemids
| -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]

[-mergelist=file1,file2,...] | [-mergefiles=merge-list-file-name]
[-sites_list=site1,site2,...] | [-sites_file=site-list-file-name]

[-file=file1] [-file=file2] [-file=file3]

[-remove_consistent] [-show_rco] [-include_bom]
[-includefoldercontent] [-delimiter=delimiter]
[-anchorfile=anchor-file-name] [-traverseditemfile=outname]
[-outfile=file-name]

[-skipItem]
{-start_creation_date=creation-date
-end_creation_date=creation-date |
[-start_modification_date=modification-date
-end_modification_date=modification-date] |
[-start_release_date=release-date
-end_release_date=release-date] }
[-sort_by=item_id | item_name | date] [-dataset_version=latest | all]
[-out_item_revs_file=output-file] [-h]

```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-function

Specifies one of the following functions:

report	Parses object information to a report file. This is the default.
merge	Parses input from a set of input files as specified by the mergelist argument.

-itemidsfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item IDs.

-item_id

Specifies a list of comma-separated values of item IDs.

-key

Specifies the keys of the items on which to report. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-itemKeyFile

Specifies the name of the file containing the keys on which to report. The file format is:

```
- key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...
- key =[keyAttr1=keyVal1][keyAttr2=keyVal2]...
- key =[keyAttr1=keyVal1][keyAttr2='keyVal2']...
```

-grmtypefile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated **grmrelations**.

-mergelist

Specifies a comma-separated list of files from individual sites in the same order as specified by the **sites_list** argument.

-mergefiles

Specifies a file containing a list of reports from individual sites in the same order as specified by the **sites_list** argument.

-sites_list

Specifies a comma-separated list of sites to be analyzed when using the **report** function. This option is ignored when using any of the date-range arguments.

-sites_file

Specifies a file containing a list of sites to be analyzed when using the **report** function. This option is ignored when using any of the date-range arguments.

-remove_consistent

Specifies that the utility does not output consistent items in the merged report.

-show_rco

Specifies that the utility displays remote checkout information. By default, this information is not displayed.

-include_bom

Specifies that the utility traverses to the end of the item revisions of **PSOccurrences**. By default, this action is not performed.

-includefoldercontent

Specifies that the utility includes the contents of folders. By default, folders are not processed.

-delimiter

Specifies the delimiter used in the output file. By default, the commercial at symbol (@) is used. Ensure the delimiter for site-based files matches the merge file input file.

-anchorfile

Specifies the output file of UIDs of revision anchors. This file is used as input to the **purge_dataset** utility.

-traverseditemfile

Specifies an output list of traversed item IDs when using the **include_bom** argument.

-outfile

Specifies an output file.

-skipItem

Specifies that items are to be skipped.

-start_creation_date

Specifies the creation *from* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **01-Jan-2007 00:00:00**. This argument is used with the **end_creation_date** argument.

-end_creation_date

Specifies the creation *to* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **Jan-2007 00:00:00**. This argument is used with the **start_creation_date** argument.

-start_modification_date

Specifies the modification *from* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **Jan-2007 00:00:00**. This argument is used with the **end_modification_date** argument.

-end_modification_date

Specifies the creation *to* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **Jan-2007 00:00:00**. This argument is used with the **start_modification_date** argument.

-start_release_date

Specifies the released *from* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **Jan-2007 00:00:00**. This argument is used with the **end_released_date** argument.

-end_release_date

Specifies the released *to* date. The date is entered in *dd-mmm-yyyy hh:mm:ss* format, for example, **Jan-2007 00:00:00**. This argument is used with the **start_released_date** argument.

-dataset_version

Specifies whether **all** or the **latest** version of the dataset needs to be reported. If this argument is not specified, the utility uses the default value of **all**.

-sort_by

Specifies one of the following attributes by which the items are processed:

- **date**
- **item_id**
- **item_name**

If this argument is not specified, the utility uses the default value of **item_id**.

-out_item_revs_file

Specifies an output file for item revisions. You can use this file as an input to the **delete_pdm_data** utility. The following is an example of an output item revision file:

```
ABC000075/A
ABC000074/A
ABC000092/A
ABN000002/A
ABN000011/A
ABN000058/A
```

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- To create reports for the latest dataset versions created between **01-jan-2007** and **01-jan-2008**, write the report file to **c:\temp\reports.txt** and write the reported item revisions to **c:\temp\itemrevs.txt**:

```
item_report -u=Tc-admin-user -p=password -g=group
  -start_creation_date="01-Jan-2007 00:00:00"
  -end_creation_date="01-Jan-2008 00:00:00"
  -outfile=c:\temp\reports.txt
  -out_item_revs_file=c:\temp\itemrevs.txt
```

migrate_organization

Allows you to create a global organization for your Multi-Site environment. Organization objects that have been duplicated across multiple sites can be changed to replicas of an owning organization site. You use the utility first to identify duplicate objects between two sites. After identifying objects that must be made replicas, use the utility make the identified duplicate objects replicas of the owning organization site objects.

Caution:

You must follow the process for migrating organization objects to successfully create your global organization. The order that you migrate objects, indicated in the process, must be followed to avoid errors when using this utility. You must run this utility at the owning site only; you can migrate objects to only one site at a time.

Note:

You must set the **IDSMD_global_dsa_sites_permitted_to_push_admin_data** preference using this utility. The effort required to roll back undesired changes is not trivial if this preference is not set correctly.

SYNTAX

```
migrate_organization [-u=user-id {-p=password | -pf=password-file} -g=group]
{-f=compare_organization -report=report-file-name | make_replica}
{ [-user_id=userid1,userid2, ... | -role_name=rolename1,rolename2, ... |
-group_name=group1,group2, ... | -person_name=person1,person2, ...] |
[-classoffile=class-name -filename=input-file ] }
{-site=remote-site-name}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the name of the function. Following functions are valid:

compare_organization

Identifies the duplicate and missing organization objects at the remote site (specified by **-site** argument) and the current site. The remote site is specified using the **-site** argument and the output is written to the file specified by the **-report** argument.

make_replica

Makes the object specified by the **-user_id**, **-group_name**, **-role_name**, or **-person_name** argument replica objects at the target (remote) site. This command must be used only at the site that is intended to be the owning site for organization objects. One class of object must be specified but only one class can be in a command. Alternatively, the **-classoffile** and **-file_name** arguments can be specified to use a file containing the objects to be replicated. See [restriction 4](#).

-report

Specifies the file name of the report generated by the **compare_organization** function.

-user_id

Specifies an ID of a user to change to a replica object at the remote site. You can specify multiple IDs separated by commas.

-group_name

Specifies the full name of a group to change to a replica object at the remote site. You can specify multiple group names separated by commas.

-role_name

Specifies the name of a role to change to a replica object at the remote site. You can specify multiple role names separated by commas.

-person_name

Specifies the name of a person to change to a replica object at the remote site. You can specify multiple person names separated by commas.

-classoffile

Specifies the class of objects contained in the input file. Valid only with the **-filename** argument. The **-classoffile** argument can specify:

- **User**

The file must contain user IDs of **User** class objects.

- **Role**

The file must contain role names of **Role** class objects.

- **Group**

The file must contain full group names of **Group** class objects.

- **Person**

The file must contain person names of **Person** class objects.

--filename

Specifies the name of a file that contains a list of identifiers for objects of a specified class. Valid only with the **-classoffile** argument.

Note:

All objects in the file must be separated by a new line character.

-site

Specifies the remote site name.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. You must set the **IDSMD_global_dsa_sites_permitted_to_push_admin_dat** preference at the remote site specifying the sites that are allowed to push organization data to the local site.
2. You must perform the premigration tasks described in *Multi-Site Collaboration* before using this utility with the **make_replica** function.
3. You must perform the premigration tasks described in *Multi-Site Collaboration* before using this utility with the **make_replica** function.
4. The **make_replica** function must be run only at the owning site of global organization data.
5. To migrate a user, the related **Person** object must be migrated first.
6. To migrate a role, for each associated **GroupMember** object, its **User** object must already be migrated.
7. To migrate a group, all of the roles related to the group must be migrated first.
8. Only a top-level group can be migrated. Specifying a sub-group fails. All sub-groups of the specified group are automatically migrated.

EXAMPLES

- Generate a report (**cvrg_cgn.txt**) file showing the duplicate objects between the current site and the remote site (**cgn**).

```
migrate_organization -u=Tc-admin-user -p=password -g=group
-f=compare_organization -report=cvrg_cgn.txt -site=cgn
```

- Make the users identified in the **cgn_user_rpl.txt** file replica objects at the remote site (**cgn**).

```
migrate_organization -u=Tc-admin-user -p=password -g=group -f=make_replica
-file=cgn_user_rpl.txt -classoffile=User -site=cgn
```

- Make the identified group objects (**ptrain_dev**, **frame_prod**) replica objects at the remote site (**cgn**).

```
migrate_organization -u=Tc-admin-user -p=password -g=group -f=make_replica
-group=ptrain_dev, frame_prod -site=cgn
```

migrate_saved_searches

Updates saved search data (which is in the form of user preferences) to the current data model. Current saved search functionality allows users to share saved searches with other users.

SYNTAX

```
migrate_saved_searches [-u=user-id {-p=password | -pf=password-file} -g=group]
-mode=upgrade | report [-owning_users=all | user-names]
[-users_of_group=all | group-names] [-file=file-name-path]
[-delete=yes | no] [-log=file-name-path] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Determines which function the utility performs. Use **report** to generate a report of legacy saved searches. Use **upgrade** to migrate the legacy saved searches to the current data model.

-owning_users

Migrates the legacy saved search data for the specified users. Use **all** to migrate the data of all users. Enter multiple user names separated by commas. If left unset, the default is **all**.

-users_of_group

Migrates the legacy saved search data for all the uses in the specified groups. Use **all** to migrate the data of all groups. Enter multiple group names separated by commas. If left unset, the default is **all**.

-file

Specifies the path and file name to which the migration report is written. Use this argument with the **report** value. The default location is *current-working-directory/migrate_saved_search_date_report.txt*.

-delete

Determines whether to delete legacy saved search data from the database after migration. Use **yes** to delete the data. The default setting is **no**.

-log

Specifies the path and file name to which any migration errors are written. Use this argument with the **upgrade** value. The default location is *current-working-directory/migrate_saved_search_date_report.log*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To migrate all the saved searches belong to the users **john** and **dave**, and to delete the user preferences after migration:

```
migrate_saved_searches -u=Tc-admin-user -p=password -g=group -mode=upgrade
-owning_users=john,dave -delete=yes
```

To generate a report of all the saved searches belonging to all users in the **design** and **manufacturing** groups:

```
migrate_saved_searches -u=Tc-admin-user -p=password -g=group -mode=report  
-users_of_group=design,manufacturing -file=C:\reports\des_mfg_mss.txt
```

To migrate all the saved searches belong to all the users in the database, and to delete the user preferences after migration:

```
migrate_saved_searches -u=Tc-admin-user -p=password -g=group -mode=upgrade  
-owning_users=all -delete=yes
```

pdx_export

Exports data in PDX format.

SYNTAX

```
pdx_export [-u=admin-id {-p=password | -pf=password-file} -g=dba]
{ -item=item-id [-rev=item-revision] | -item_key=item-key -sitename=target-site}
[-optionset=transfer-option-set][-reason=reason-description]
[-immediate= {True | False} ] [-notify= {True | False} ]
[-emailaddrs=email-address1, email-ddress2, email-addressn]
[-revisionrule=revision-rule-name] [-bomlevel=level-of-BOM-structure]
[-vendors=vendor-name1, vendor-name2, vendor-namen] [-fileoutput-file-name]
[-usegs= {True | False} ]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. The group value must be **dba** to run this utility.

-item

Specifies the identifier for the item object to export. This argument is mutually exclusive with the **-item_key** argument.

-rev

Specifies the revision of the item object to export. You can use this argument only when you include the **-item** argument.

-item_key

Specifies the a string identifier containing attributes that identify the item object to export. This argument is mutually exclusive with the **-item** argument.

-sitename

Specifies the name of the importing (target) site.

-optionset

Specifies the name of the transfer option set to use for the export.

-reason

Allows you to type a description of the export purpose. This argument is limited to 240 characters. If you enter more than 240 characters, the argument is truncated.

-immediate

Specifies whether to schedule the export or perform it immediately. Valid values are **True** or **False**.

-notify

Specifies whether to send a e-mail notification to the users specified in the **-emailaddrs** argument. Valid values are **True** or **False**.

-emailaddrs

Specifies a comma delimited list of e-mail addresses that are notified of the export.

-revisionrule

Specifies the name of the revision rule to use for the export.

-bomlevel

Specifies the level of BOM to traverse for the export.

-vendors

Specifies a list of vendor names used to filter the content of the exported data. Only objects associated with a vendor in this list are included in the export file. If this argument is not specified, objects associated with any vendor are included.

-file

Specifies the name of the output file containing the PDX export data. If the file exists, it is overwritten.

-usegs

Specifies whether to use Global Services to perform the export.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

You must be logged on as a member of the **dba** group.

plmxml_export

Exports objects from Teamcenter in the PLM XML format. If there are files for export, as determined by transfer mode, a directory is created and the files are exported into that directory. The directory is named using the specified file name without the **.xml** extension.

This utility is also used to extract file information from the database into a PLM XML file to prepopulate the FMS server caches (FSCs).

For more information, see the **load_fccache** and **fscadmin** utilities.

Caution:

Do not use this utility to export organization objects from sites if the same organization objects are shared throughout a global organization.

Warning:

The Business Modeler IDE application maintains most of the objects that affect the data model. Do not export the following objects. Importing the generated PLM XML file using the **plmxml_import** utility may cause data accuracy issues.

ActivityTypeDef	GRMRule	PropBusinessOperation
AliasTypeDef	HideTypeRule (Type-Display rule)	PropertyRule
AppearanceGroupTypeDef	IdContextRule	RelationTypeDef
ApplicationInterface	ImanTypeDef	StatusTypeDef
CannedMethodRule	ItemTypeDef	StorageMediaTypeDef
ChangeTypeDef	ListOfValues	ToolTypeDef
CompoundPropDefRule	NameFieldRule	TypeBusinessOperation
DatasetTypeDef	NamingRule	UOMTypeDef
DeepCopyRule	NoteTypeDef	ViewTypeDef
Extension	OccurrenceTypeDef	WorkAreaTypeDef
FolderTypeDef	OperationTypeDef	
FormTypeDef	ProcessTypeDef	

SYNTAX

```
plmxml_export [-u=user-id {-p=password | -pf=password-file} -g=group]
              -xml_file=xml-file-name -transfermode=transfermode-name
```

```
{[-item=item-id | -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
[-rev=item-revision-id] [-export_bom=yes | no]
[-rev_rule=revision-rule] [-svrule=saved-variant-rule-name [:rev-id:rule-id]]
| -class=class-name | -type=type-name
| -ics_class=ics-class-name| -imantypedef=iman-type
| -uid=uid-of-object | -foldername=folder-to-export}
| [-template=workflow-template-name] {[-template_stage=workflow-template-stage]
[-template_classification=workflow-template-classification]}
[-locales=language-ID]
[-session_options=opt1:val11[:val12...:val1N][;opt2:val21[:val22...:val2N]]...
[:optN:valN1[...:valNN]]}
[-query=query-name{user-entry-name1:val1[,user-entry-name2:val2]...[,user-entry-nameN:valN]}]
[-log=log-file-location] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-xml_file

Specifies the full path of the file to which the data is exported.

-transfermode

Specifies the name of the transfer mode used to export the objects. This transfer mode specifies the traversal rules, filter rules, and property sets to be used for export. It determines what is exported from the system. If not specified, a default transfer mode is used. If **-transfermode** is set to **justDatasetsOut**, you must specify a revision ID using the **-rev** argument.

-item

Specifies the ID of the item to be exported.

-key

Specifies the keys of the items to be exported in PLM XML format. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see Multifield keys.

-rev

Specifies the revision ID of the item to be exported. If not specified, the configured revision (either specified or default) is exported for the item.

-rev_rule

Specifies the revision rule applied to export a BOM. This option is also used to determine the configured revision if the **-rev** option is not specified.

If this option is not specified, the default revision rule is applied, as specified in the **TC_config_rule_name** preference.

-svrule

Specifies a saved variant rule to use when configuring the exported BOM. Optionally, specify a particular revision of a variant rule object by providing its revision ID and rule ID separated by colons. Use of the variant rule revision is supported for only **Cfg0VariantCriteria**, a revisable subclass of the **VariantRule** object.

-class

Exports all instances of a given class.

Note:

- You cannot use the **-class** argument to export scope rules (**TransferModes**, **ClosureRules**, **FilterRules**, **PropertySets**, **ActionRules**, and **TransferOptionSets**). To export rules, use the **tcxml_export** utility.

- To view the persistent object manager (POM) schema, open the **Classes** view in the Business Modeler IDE.
- If **ListOfValues** and **BusinessRule** are specified, all instances of subclasses of the specified class are also exported.

Options to export organization information using **-class**:

Option	Description
Person	Export all persons
User	Export all users
Role	Export all roles
Group	Export all groups
GroupMember	Export all group members
POM_imc	Export all sites
ImanVolume	Export volumes
ListOfValues	Export all LOVs (lists of values)
ListOfValuesString	Export string LOVs
ListOfValuesDate	Export date LOVs
ListOfValuesDouble	Export double LOVs
ListOfValuesInteger	Export integer LOVs
ListOfValuesChar	Export char LOVs
ListOfValuesTag	Export reference LOVs

Options to export workspace objects using **-class**:

Option	Description
Item	Export all instances of items
ItemRevision	Export all instances of item revisions
Folder	Export all instances of folders
Form	Export all instances of forms
Dataset	Export all instances of datasets
Alias	Export all instances of alias

Option	Description
EPMJob	Export all instances of workflow jobs
PSBOMView	Export all instances of BOM views
PSBOMViewRevision	Export all instances of BOM view revisions
Tool	Export all instances of tools

Options to export business rules using **-class**:

Option	Description
BusinessRule	Export all business rules
NameRule	Export all naming rules
NameField	Export all naming fields
HideTypeRule	Export all hide type rules of tools
ImanCompoundPropDef	Export all compound property rules of tools
ImanGRM	Export all GRM rules
TypeCannedMethod	Export all action rules

Other options using **-class**:

Option	Description
FormTypeDef	Export all form type definitions
ImanType	Export all instances of Teamcenter types
NoteType	Export all note types
PSViewType	Export all view types
UnitOfMeasure	Export all defined unit of measures
TaskType	Export all defined task s
PSOccurrenceType	Export all occurrence types

-type

Exports all instances of a given type.

-ics_class

Exports the specified classification class, if it exists.

-imantypedef

Exports the definition of specified type.

Options and their results for **-imantypedef**:

Option	Description
ListOfValues	Export list of values
ImanQuery	Export saved queries
Tool	Export tool definitions
TaskType	Export defined task types
IdContext	Export identifier contexts
Status	Export defined status
StorageMedia	Export storage media
Note	Export PS occurrence note types
UnitOfMeasure	Export unit of measures defined
Occurrence	Export occurrence types
View	Export view types
RevisionRule	Export revision rules for PS configuration
Alias	Export alias and its types
Identifier	Export identifier and its types
WorkArea	Export workarea
OP	Export operation
ChangeTypeData	Export changeid/changetype
Dataset	Export dataset type definition
ImanType	Export all Teamcenter types
ImanRelation	Export relations
AppearanceGroup	Export appearancegroup types

-export_bom

Specifies that the BOM is exported. This argument must be used in conjunction with the **-item** argument.

-uid

Exports the object specified by the unique ID (UID).

-foldername

Exports the specified folder, if it exists.

-template

Specifies the name of the exported workflow template.

-template_stage

Specifies the stage type of the exported workflow template. This argument is used with the **-template** argument. Valid values are **OBSOLETE_STAGE**, **UNDER_CONSTRUCTION_STAGE**, or **AVAILABLE_STAGE** (default). If this option is not specified, the default value is used.

-template_classification

Specifies the classification type of the exported workflow template. This argument is used with the **-template** argument. Valid values are **TASK_TEMPLATE** and **PROCESS_TEMPLATE** (default). If this option is not specified, the default value is used.

-session_options

Specifies generic session options using colon separated name-value pairs. Name-value pairs can contain multiple values paired with a name. Options are delimited by commas. Multiple values in pairs are delimited by semicolons.

-query

Specifies a saved query to use when exporting an object. *query-name* is the saved query name. User entry names are attribute-value pairs delimited by colons. Multiple pairs are delimited by commas. **-query** is intended for cases where a single object is returned.

-locales

Specifies the languages for the export. Separate multiple languages by commas. The language IDs should follow the standard Java locale naming conventions (for example, **en_US**). If no locales are specified for export, the database scalar value (primary attribute) is exported to PLM XML scalar fields.

-log

Specifies the full path of the export log file. If this option not specified, the log file is created in the **\$TC_TMP_DIR** directory. If **\$TC_TMP_DIR** is not defined, the log file is created in the system temporary directory (**C:\TEMP** on Windows or **/tmp** on Linux).

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- Siemens Digital Industries Software recommends you do *not* use the following transfer modes with this utility:

```
BOMwriterExport
JTDataExportDefault
TIEPDExportDefault
TIEUnconfiguredExportDefault
PLMXMLAdminDataExport
```

- Do not use this utility to export organization objects from sites if the same organization objects are shared throughout a global organization.

EXAMPLES

- The following command exports item **ABC00001**, revision **A**, to a PLM XML file using the **toPrimeSupplier** transfer mode context. The default revision rule is applied.

```
plmxml_export -u=Tc-admin-user -p=password -g=group
-xml_file=abc00001_A.xml -item=ABC00001 -rev=A
-transfermode=toPrimeSupplier
```

- The following command exports item **ABC00002**, revision **A** to a PLM XML file using the **toEnterprise** transfer mode context by applying the specified revision rule and saved variant rule to apply to the BOM window:

```
plmxml_export -u=Tc-admin-user -p=password -g=group
-xml_file=abc00002_A.xml
-item=ABC00002 -rev_rule="Latest Released" -svrule="AlphaRelease"
-transfermode=toEnterprise
```

- The following command exports all users to an XML file using the default context to export the data to PLM XML format:

```
plmxml_export -u=Tc-admin-user -p=password -g=group -xml_file=tcusers.xml
-class=User
```

- The following command exports an object specified by the UID and uses the default context to export the data to PLM XML format. The UID should be a unique identifier in Teamcenter:

```
plmxml_export -u=Tc-admin-user -p=password -g=group -xml_file=myobj.xml
-uid="QRw4LZ0g1YomJAAAAAAAAAAAAAAAA"
```


- The following command creates a PLM XML file containing all external file references associated with the top level item selected for cache prepopulation.

```
plmxml_export -u=Tc-admin-user -p=password -g=group -item=ITEM -rev=A
-export_bom=yes -transfermode=justDatasetsOut -out=tickets.plmxml
```

- The following command exports the item with ID **item1** to the PLM XML file **item1.xml**. The French and German translations of the localized properties on **item1** that are identified for export (from the property set) are also exported to text elements.

```
plmxml_export -u=Tc-admin-user -p=password -g=group -item=item1
-locales=fr_FR,de_DE -xml_file=item1.xml
```

- The following command exports all partitions to an XML file using the default context to export the data to PLM XML format. In this example of exporting generic session options, **-session_options** is used to export selective partition scheme types.

```
plmxml_export -u=Tc-admin-user -p=password -g=group -xml_file=tcusers.xml
-session_options=
Multiple_partition_schemes:Ptn0SchemeFunctional;Ptn0SchemePhysical
```

- The following command uses the saved query "Design Element" to export the element with a user entry value of "Design Element Name". Double quote symbols surround the **-query** option value because the value includes spaces.

```
plmxml_export -u=Tc-admin-user -p=password -g=group -query="Design
Element{Design Element Name:DE_QRY}" -xml_file=1234.xml
-transfermode=4GDPIEDataExportDefault
-rev_rule="Working; Any Status"
```

plmxml_import

Imports objects to Teamcenter from a specified PLM XML file. In cases where a transfer mode manages the import, the utility looks for files in the path specified by the **xml_file** argument. This utility is also used to import workflow templates. If the PLM XML file being imported contains translations of localizable properties in multiple languages, the translations of the supported languages are imported into the database.

Caution:

Do not use this utility to import organization objects to sites if the same organization objects are shared through a global organization.

Warning:

The Business Modeler IDE application maintains most of the objects that affect the data model. Do not import the following objects. Importing the generated PLM XML file using the **plmxml_import** utility may cause data accuracy issues.

AliasTypeDef	GRMRule	ProcessTypeDef
ApplicationInterface	HideTypeRule (Type-Display rule)	PropBusinessOperation
AppearanceGroupTypeDef	IdContextRule	RelationTypeDef
ActivityTypeDef	ImanTypeDef	StatusTypeDef
CannedMethodRule	ItemTypeDef	StorageMediaTypeDef
ChangeTypeDef	ListOfValues	ToolTypeDef
CompoundPropDefRule	NamingRule	TypeBusinessOperation
DatasetTypeDef	NameFieldRule	UOMTypeDef
DeepCopyRule	NoteTypeDef	ViewTypeDef
Extension	OccurrenceTypeDef	WorkAreaTypeDef
FolderTypeDef	OperationTypeDef	
FormTypeDef	PropertyRule	

SYNTAX

```
plmxml_import [-u=user-id {-p=password | -pf=password-file} -g=group]
-xml_file=name-of-xml-file -transfermode=transfermode-name
[-log=log-file-name] [-import_mode=overwrite | ignore]
[{-apply_template | -ignore_originid}] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-xml_file

Specifies the full path of the file name from which the data is imported.

-transfermode

Specifies the name of a transfer mode used to import the objects. This transfer mode specifies the traversal rules, filter rules, and property sets to be used for import. It determines what is imported from the system. If not specified, a default transfer mode is used. In addition, the following transfer mode values can be applied to import workflow templates.

workflow_template_import

Use to create a new template. If the template already exists in the database, the command is ignored.

workflow_template_overwrite

Use to overwrite an existing workflow template. The version existing in the database is overwritten.

by the imported version. If the workflow template does not already exist, a new workflow template is created.

-log

Specifies the full path of the import log file. If this option not specified, the log file is created in the **\$TC_TMP_DIR** directory. If **\$TC_TMP_DIR** is not defined, the log file is created in the system temporary directory (**C:\TEMP** on Windows or **/tmp** on Linux).

-import_mode

Specifies the mode in which import is handled for PLM XML Import/Export configuration objects. In **overwrite** mode, objects that already exist in the database are overwritten. In **ignore** mode, the imported object is ignored if the imported object already exists in the database.

The classes that function with this argument are:

Teamcenter class name	SDK class name
TransferMode	plmxml60::TransferMode
ClosureRule	plmxml60::ClosureRule
PropertySet	plmxml60::PropertySet
Filter	plmxml60::FilterRule
PIEActionRule	Exported as UserData under plmxml60::TransferMode
Person	plmxml60::Person
TCCalendar	plmxml60::Calendar
User	plmxml60::User
Group	plmxml60::Organisation
Discipline	plmxml60::Discipline
Role	plmxml60::Role
POM_imc	plmxml60::Site
RevisionRule	plmxml60::RevisionRule
ListOfValues	plmxml60::ListOfValues
ImanQuery	plmxml60::SavedQueryDef

-apply_template

When you specify the **workflow_template_overwrite** transfer mode, and the imported workflow template contains changes from the existing workflow template, this argument applies those changes to all active workflow processes based on the workflow template.

When you import templates from a Teamcenter version prior to 10.1, do not use the **-apply_template** argument. If you do, Teamcenter does not successfully import the template.

For more information about how the workflow template changes are applied, see *Workflow Designer on Rich Client*.

This argument must be used with the **workflow_template_overwrite** transfer mode.

The **-apply_template** and **-ignore_originid** arguments are mutually exclusive.

-ignore_originid

Prevents the check of the **origin_id** property and forces the imported workflow template to overwrite the current one. Changes to active workflow processes are not applied.

The **-apply_template** and **-ignore_originid** arguments are mutually exclusive.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- Siemens Digital Industries Software recommends you do *not* use the following transfer modes with this utility:

JTDataImportDefault
TIEImportDefault

- Do not use this utility to export organization objects to sites if the same organization objects are shared through a global organization.

EXAMPLES

- The following command imports all objects in the **abc.xml** file using the default context:

```
plmxml_import -u=Tc-admin-user -p=password -g=group -xml_file=abc.xml
```

If errors are detected during the import operation, a log file named **abc_log.txt** is created.

- The following command imports the **wkf_templates.xml** workflow template without overwriting existing templates:

```
plmxml_import -u=Tc-admin-user -p=password -g=group
-xml_file=wkf_templates.xml
-transfermode=workflow_template_import
```

- The following command imports the **wkf_templates.xml** workflow template that overwrites the existing **wkf_templates.xml** workflow template, creating an updated version of the template in the database:

```
plmxml_import -u=Tc-admin-user -p=password -g=group
-xml_file=wkf_templates.xml
-transfermode=workflow_template_overwrite
```

- The following command imports the **wkf_templates.xml** workflow template that overwrites the existing **wkf_templates.xml** workflow template, creating an updated version of the template in the database, and applies all changes from the imported version to all active workflow processes:

```
plmxml_import -u=Tc-admin-user -p=password -g=group
-xml_file=wkf_templates.xml
-transfermode=workflow_template_overwrite -apply_template
```

plmxml_tm_edit_xsl

Lists, exports, attaches, or detaches an **.xslt** file to a given transfer mode.

If you are using multiple **.xslt** files, you must run the utility for each file. The files are applied in the order in which you include them in and run the utility.

SYNTAX

```
plmxml_tm_edit_xsl [-u=user-id {-p=password | -pf=password-file} -g=group]
-transfermode=transfermode-name
-action= | list | export | attach | detach | detach_all
-xsl_file=xslt-filename
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-transfermode

Specifies the transfer mode to which the **.xslt** file is exported, attached, or detached.

-action

Performs one of the following actions on the transfer mode:

list

Lists all **.xslt** files associated with the transfer mode.

export

Exports the **.xslt** file to the operating system.

attach

Attaches the **.xslt** file to the transfer mode.

detach

Detaches and removes the **.xslt** file from the transfer mode.

detach_all

Detaches and removes all of the **.xslt** files from the transfer mode

-xslt_file

Specifies the **.xslt** file.

This option is required if the **-action** option is set to **export**, **attach**, or **detach**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

restore

Accepts input objects of type **Item** and 4GD objects of type **MdlOApplicationModel** which were previously archived from the current site. This utility restores the objects to the current site along with their dependent objects with ownership privileges. Run this utility from the owning site. Restoring is a pull operation from the archive site to the owning site. (The companion *archive* operation is a push operation from the owning site to the archive site.)

SYNTAX

```
restore -u=user-ID {-p=password | -pf=password-file} -g=group
```

```
-f={list | perform} [-item_id=itemid] [-rev=revision]
[-uidfile=data_file] [-itemidsfile=data_file] [-key=key] [-itemKeyFile=data_file]
[-include_bom] [-report_file=report_filepath] [-4gd_id]
[-verbose] [-class=wso_class_name] [-classoffile=class_name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

list Performs a dry run and reports the objects that would be restored from the database.

perform Performs a restore operation and reports the objects that are restored from the database.

-item_id

Specifies the Item ID of the item to process.

-rev

The revision of the item to process.

-uidfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item UUIDs of items to process.

-itemidsfile

Specifies a data file containing comma-separated values (CSVs) or carriage return/line feed separated item IDs of items to process.

-key

Specifies a comma-separated list of items or 4GD key name and value pairs. If the key is for a 4GD object, the related 4GD class must also be specified with **-class=**. Use the following format:

keyAttr1=keyVal1[,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

Item example:

item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1

4GD example:

4gd_id=CD000001_ID,object_name=CD000001_Name,object_desc=CD000001_Desc

-itemKeyFile

Specifies the name of the file containing the list of Item or 4GD object key strings to process. The file format is:

keyAttr1=keyVal1[,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

If the list has 4GD object key strings, the related 4GD classes must also be specified with **-classoffile=**.

File list with item IDs:

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

File list of item IDs with revisions:

```
item_id=s1005,rev_id=B
item_id=s1005-1,rev_id=A
item_id=s1005-2,rev_id=C
```

File list of 4GD objects with **-classoffile=**:

```
4gd_id=CD_Export_001
4gd_id=CD00001_ID,object_name=CD00001_Name,object_desc=CD00001_Desc
```

-include_bom

Optionally includes assembly components of the BOM to all levels for processing. This option cannot be used with 4GD objects.

-report_file

Specifies that a report be saved to the given file path. If the *report_filepath* argument is not given, the archive report will be created in the folder containing the utility logs.

-4gd_id

Specifies a 4GD object identifier or 4GD object pattern to process. The **-class** or **-classoffile** argument must be specified with the **-4gd_id** argument.

The utility maps the **-4gd_id** argument to the corresponding unique ID of the 4GD class. For example:

```
Class=Cpd0CollaborativeDesign, 4gd_id=CD_100
```

A 4GD partition object uses multifield key attributes by default. Consequently, using **-4gd_id** for partition objects may result in the restoring of multiple objects.

-verbose

Includes additional detailed messages in the utility's output.

-class (cl)

Specifies the Teamcenter class of the object specified by the **-4gd_id** argument.

-classoffile (cof)

Specifies the 4GD Teamcenter class of the objects contained in the input file. Valid on with the **-itemKeyFile** argument.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment..

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

This utility accepts input objects of type **Item** and 4GD objects of type **MdlOApplicationModel** only.

The utility removes all replicated and published information associated with input objects and further transfers the input objects and their dependents to the archive site with ownership privileges.

You cannot restore an older revision if the latest revision is still archived.

EXAMPLES

- Generate a dry run report of an item ID and its dependent objects that would be restored:

```
restore -u=Tc-admin-user -p=password -g=group
        -f=list -verbose -item_id=ABC_101
```

Generate a dry run report for a given input file of item ID strings that would be restored:

```
restore -u=Tc-admin-user -p=password -g=group
        -f=list -itemidsfile=item_archive.txt -verbose
```

Perform a restore and generate a report for a given item ID:

```
restore -u=Tc-admin-user -p=password -g=group
        -f=perform -verbose -item_id=ABC_101
        -report_file=c:\temp\myreport.txt
```

step_export

Exports Teamcenter data from the database to STEP-compliant physical files.

There are two ways to use this utility: single-line and batch mode. The single-line method uses the **-item=***item-id* argument and other optional arguments to export one object at a time; batch mode uses the **-i=***input-file* argument and an input file to export several objects at once.

SYNTAX

```
step_export [-u=user-id {-p=password | -pf=password-file} -g=group]
  {-i=input-file
  | [-item=item-id | -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
  [-item_rev=item-rev-id] [-ds=dataset] [-rel=relation-name]]}
  -fmt=AP203 | AP214 | IMAN [-full_assembly]
  [-all_ds_versions] [-f=file-name] [-cmt=comments] [-h] [-v]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-i

Specifies the input file containing the list of objects to export. The complete path name must be provided.

-item

Specifies the item ID of the item being exported.

-item_rev

Specifies the item rev ID of the item revision being exported.

-ds

Specifies the dataset being exported.

-rel

Specifies the name of the Teamcenter relation containing the dataset.

-key

Specifies the keys of the items to export. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-fmt

Specifies the data output format: **AP203**, **AP214**, or **IMAN**.

-full_assembly

Specifies that the full assembly, including product structure and all component parts that constitute the assembly, are exported.

-all_ds_versions

Specifies that all version of the datasets are included in the export.

-f

Specifies the output file name. The complete file specification (full path and file name) must be supplied unless the desired location is the current working directory.

-cmt

Describes the data being exported. This comment is placed in the **file_description** section of the output file.

-v

Runs utility in verbose mode, displaying maximum amount of information. Typically, nonverbose utility sessions only display error messages.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter* and the following:

`$ROSE_DB/* .rose`

.rose files are STEP schema files used by the STEP Translator. The **step_export** utility must be able to write these files in this directory.

RESTRICTIONS

Either the **-item=***item-id* or the **-i=***input-file* argument must be supplied.

EXAMPLES

To export several objects in batch mode (using an input file), perform the following:

1. Create an input file and add one line for each object you want to export in the following format:

Note:

Ensure that you separate each argument with a semicolon (;) and put each object on its own line.

```
-item=item-id; -item_rev=item-rev-id; -ds=dataset; -rel=relation-name
-item=item-id; -item_rev=item-rev-id; -ds=dataset; -rel=relation-name
```

2. Run the **step_export** utility using the **-i=***input-file* argument:

```
$TC_ROOT/bin/step_export -u=Tc-admin-user -p=password -g=group -i=input-file
```

step_import

Imports product information from STEP-compliant physical files into the Teamcenter database.

SYNTAX

```
step_import [-u=user-id {-p=password | -pf=password-file} -g=group]
{-f=file-name | -i=input-file} [-h] [-v]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies a single STEP file. The complete file specification (full path and file name) must be supplied unless the file is in the current working directory.

-i

Specifies the input file containing list of STEP files to batch process. The complete file specification (full path and file name) must be supplied unless file is in the current working directory.

-h

Displays help for this utility.

-v

Runs utility in verbose mode, displaying maximum amount of information. Typically, nonverbose utility sessions only display error messages.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter* and the following:

`$ROSE_DB/* .rose`

.rose files are STEP schema files used by the STEP Translator. The **step_export** utility must be able to write these files in this directory.

RESTRICTIONS

Either the **-f=file-name** or the **-i=input-file** argument must be supplied.

sync_form_util

Creates or modifies the **ParticipatingSitesForm** type used to control replication of released assemblies to designated sites.

SYNTAX

```
sync_form_util [-u=user-name {-p=password | -pf=password-file} -g=group]
{-item_id=root-item -rev=
-f={create | add | mv} [forminfo
-project=project-id -sitelist={site1, site2, ..., siten}}
[-h]
```

ARGUMENTS

Note:

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

If this argument is not used, the system assumes the *user-ID* value is the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

Caution:

For HTTP enabled sites, remote site operations log on using the default group for the user supplied with the **-u** argument. Any value supplied with the **-g** argument is ignored.

-item_id

Specifies the top item in the structure context object (SCO) assembly.

-rev

Specifies the top item revision used to configure the structure.

-f

Specifies the function to perform.

create

Creates and attaches the **ParticipatingSites** form.

add

Adds the site(s) specified in the **-sitelist** argument to the **SiteList** attribute of the **ParticipatingSites** form.

mv

Removes the site(s) specified in the **-sitelist** argument from the **SiteList** attribute.

forminfo

Retrieves the **ParticipatingSites** form information.

-project

Specifies the name of the project associated with the participating sites.

-sitelist

Specifies a comma delimited list of sites to receive the replicated SCO assembly.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

EXAMPLES

Note:

Required logon information is omitted from the following examples.

- A participating site form for the **B1_Y** project:

```
sync_form_util -item_id=ABC00002 -rev="Latest Released" -f=create -project=B1_Y  
-sitelist=CologneEng, AnnArborEng, CambrigeEng
```

sync_on_demand

Synchronizes or reports the synchronization state of a specified component, assembly, or object. The synchronization state indicates whether the replica is up-to-date and whether any object that has been added to the owning site has been replicated by the site running the utility.

For 4th Generation Design (4GD) objects, this utility can be used to report the synchronization state of objects between the central site and the local site. The synchronization state of 4GD objects is not affected.

The site that runs the **sync_on_demand** utility and the site that owns the component, assembly, or object being synchronized must both be instances of Teamcenter that support on-demand synchronization. If an assembly contains components from sites that do not support on-demand synchronization, the state of those components are reported as **unknown**.

Component synchronization allows you to determine the state of, or synchronize, all objects associated with the specified revision, such as BVR and attachments.

Assembly synchronization allows you to determine the state of, or synchronize, an entire assembly.

Object synchronization allows you to determine the state of individual objects, such as a dataset or form. You can also select item or item revisions for object synchronization, however the state or objects associated with the item or item revision is not reported or affected.

This utility uses the IDSM process at the remote replica's owning site to accomplish the report task and remote import to accomplish the synchronization.

SYNTAX

sync_on_demand [-u=*user-id* {-p=*password* | -pf=*password-file*} -g=*group*]

-f={*sync* | *report* [-uid_report=*uid-report-file-name*]}

-type={*object* | *component* | *assembly* | *4GD*

-recipe=*recipe-name* -site=*site-name-for-4gd-compare*}

[-rev_rule=*revision-rule* | -rev=*rev-id*] [-assy_level=*number*]

[-item_id=*item-id* | -key=keyAttr1=keyVal1 [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

| *template* | -folder=*folder-name* | -name=*wso-name* | -filename=*file-name*

| -itemKeyFile=*file-name*]

[-class=*wso-class-name* | -classoffile=*class-name*]

[-exclude=*relation-type1* [-exclude=*relation-type2* ...]]

[-include=*relation-type3* [-include=*relation-type4*...]]

[-exclude_folder_contents] [-exclude_protected_objects]

[-exclude_protected_comp] [-batch_size=*number-of-objects-per-batch*]

`[-report_file=report-file-name] [-error_report=error-file-name]`
`[-separator=uid-separator-characters] [-h]`

ARGUMENTS

Note:

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the function to perform. Valid values are **sync** or **report (rep)**. Where **sync** performs a synchronization on the specified object, component, or assembly, and **report** returns the synchronization state.

-uid_report (ur)

Returns a report on the specified object, component, or assembly that contains the UIDs of objects enclosed within square brackets ([]) by default. The characters used to enclose the UIDs are configurable through the **-separator** argument. This file can be processed by a custom script to

create an input file that the **data_share** utility uses to import objects from a remote site. Using UIDs for remote import increases performance by eliminating the remote query required to determine an item or item revision UID.

-separator (sep)

Designates the start and end characters used to enclose the UID of objects in a **uid_report** file. If this argument is not specified, the UIDs are enclosed in square brackets ([]).

-type

Specifies the type of Teamcenter object on which to perform the function. Valid values are:

- **object (obj)**
- **component (comp)**
- **assembly (assy)**
- **4GD**

Reports the status of 4GD object synchronization between the target site and the local site. Synchronization status (**-f=report**) is the only supported function for this type of object. See *Restrictions*.)

-recipe

Specifies the 4GD recipe name (**mdl0SubsetDefinition**) used for reporting 4GD synchronization state between sites. This argument is valid only with the report function (**-f=report**) argument and 4GD type (**-type=4GD**) argument. The recipe must exist and be identical at the sites being compared. The **-site** argument is also required and all other arguments are ignored.

-site

Specifies the central (target) site name. This site is compared to the local site (site where you are running the utility) for 4GD synchronization state reporting. This argument is only valid with the report function (**-f=report**) argument and 4GD type (**-type=4GD**) argument. The **-recipe** argument is also required and all other arguments are ignored.

-rev_rule

Specifies the name of the revision rule used to perform the synchronization or report function. This name must specify an existing revision rule at the local site. This value is passed to the owning site where it is used to determine the item revision to report status for or synchronize. You must supply this argument if the target object is an assembly.

-rev

Specifies the revision ID of the revision to report the status of or synchronize.

-assy_level (al)

Specifies the number of levels of the assembly to report. This argument is valid only for the report function.

-item_id (item)

Specifies the ID or template of items to report the status of or synchronize. Mutually exclusive with the **-name**, **-folder**, **-key**, **-itemKeyFile**, and **-filename** arguments.

-key

Specifies the keys of the items to export. Use the following format:

```
keyAttr1=keyVal1 ,keyAttr2=keyVal2... ,keyAttrN=keyValN
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

Mutually exclusive with the **-name**, **-folder**, **-filename**, **-itemKeyFile**, and **-item_id** arguments.

-folder (fl)

Specifies the folder that contains the object to report the status of or synchronize. If the folder is not unique, the first folder found that matches this value is used. Mutually exclusive with the **-name**, **-filename**, **-key**, **-itemKeyFile**, and **-item_id** arguments.

-name

Specifies the name of a single workspace object to be precessed. If not an item, use the **-class** option to specify the class of the object. Mutually exclusive with the **-folder**, **-filename**, **-key**, **-itemKeyFile**, and **-item_id** options.

-filename (fn)

Specifies the name of the input file containing IDs or names of objects to report the status of or synchronize. Mutually exclusive with the **-name**, **-folder**, **-key**, **-itemKeyFile**, and **-item_id** arguments. If the input file contains names, the **-classoffile** argument is required.

-itemKeyFile

Specifies the name of the output file containing the keys to export, for example:

```
-itemKeyFile=itemids.txt
```

An example of the file format is:

```
item_id=export_001
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

Mutually exclusive with the **-name**, **-folder**, **-filename**, **-key**, and **-item_id** arguments.

-class (cl)

Specifies the Teamcenter class of the object specified by the **-name** argument. This argument is valid only with the **-name** argument. If not specified, **Item** is the default class.

-classoffile (cof)

Specifies the class of objects in the input file. This argument is valid only with the **-filename** argument. If not specified, **Item** is the default class. **Folder** is a valid class for synchronization.

-exclude

Excludes the specified relation type. This argument may be given multiple times and must use the database name (not the display name) of the relation type.

Tip:

For best tcxml performance, use a closure rule for relation instead of include/exclude.

-include

Includes the specified relation type. This argument may be given multiple times and must use the database name (not the display name) of the relation type. Use this argument to force the inclusion of a relation type that may have been excluded during the last export.

Tip:

For best tcxml performance, use a closure rule for relation instead of include/exclude.

-exclude_folder_contents (efc)

Excludes the contents of a folder. Intended for use with NX part families where family members are stored in a folder that is related to the item.

-exclude_protected_object (epo)

Excludes export protected objects.

-exclude_protected_comp (epc)

Excludes export protected components. This argument is valid only when the **-type** argument is set to **assembly**.

-batch_size (bs)

Specifies the number of objects per batch; a new process is created per batch. Default batch size is 1000. Must be a positive integer. This is useful when processing thousands of objects, because it helps avoid memory and disk space shortage problems.

-report_file (rf)

Generates a report that is output to the specified file.

If the function specified is **report**, the report contains the synchronization state of each object, component, or components of an assembly.

If the function specified is **sync**, the report shows all successful imports and any errors that occur during the synchronization.

If the function specified is **report** and this argument is not supplied, the report is displayed in a shell (**stdout**).

-error_report (erep)

Generates a error report that is output to the specified file. This file contains only error information. It provides no synchronization information. If a file name is not supplied, the report is displayed in a shell. This argument is normally used with the **-sync** argument to provide error information during the synchronization process.

-h

Displays help for this utility.

RESTRICTIONS

Identical 4GD search recipes must exist at both the local and central sites. The 4GD on demand synchronization feature requires the **-f=report**, **-recipe**, and **-site** arguments and supports the **-report_file** argument. All other arguments are not supported and are ignored if supplied.

EXAMPLES

Note:

Required logon information is omitted from the following examples.

- To synchronize an assembly and output any errors that occur to **stdout**:

```
sync_on_demand -f=sync -type=assembly -rev_rule="Latest Working"
-item_id=Item100/A -error_report=err_rep.txt
```

The assembly components are synchronized and errors are output to **stdout**.

- To generate a report called **assy_sync.rpt** for the **Item100/A** assembly:

Enter the following command on a single line:

```
sync_on_demand -f=report -type=assy -rev_rule="Latest Working"
-item_id=Item100/A -report_file=assy_sync.rpt
```

An example of the contents of the **assy_sync.rpt** file on the owning site:

Component LMD	Owning Site	Sync State	Master LMD	Replica
Item100/A	Site1		09/25/05	09/25/05
Item100/A-view	Site1		09/25/05	09/25/05
Item101/B	Site2	out of date	10/25/05	09/25/05
Item101/B-view	Site2	out of date	10/25/05	09/25/05
Item102/A	Site2		09/25/05	09/25/05
Item102/A-view	Site2		09/25/05	09/25/05

Item103/C	Site1		09/25/05	09/25/05
Item104/B	Site3	unknown		

The report contains components up to the highest level that is out-of-date within a branch. For example, in the sample report, the BVR of component **Item101/B** is out of date and no further expansion is done to show its children in this branch. The branch with **Item102/A** has an up to date BVR so it is expanded until a leaf node is encountered or until a BVR with an out-of-date or unknown status is found.

- To generate a report called **comp_sync.rpt** for the **Item100** component:

Enter the following command on a single line:

```
sync_on_demand -f=report -type=comp -rev_rule="Latest Working"
               -item_id=Item100/A -report_file=comp_sync.rpt
```

An example of the contents of the **comp_sync.rpt** file on the owning site:

Object String Replica	Type	Relation	Owning Site	Sync State	Master LMD	LMD
Item100 09/25/05	Item			out of date	10/25/05	
Item100 09/25/05	Item Master	IMAN_master_form	Site1		09/25/05	
Item100-view Item100/A 09/25/05	BOMView Item Revision	Revision	Site1	out of date	10/25/05	
Item100/A 09/25/05	Rev Master	IMAN_master_form	Site1		09/25/05	
Item100/A-spec 09/25/05	UGMASTER	IMAN_specification	Site1	out of date	10/25/05	
Item102/B-ref Item100/A-view 09/25/05	Text BOMViewRev	IMAN_reference	Site2 Site1	unknown	09/25/05	

- To create a file (**uids_list.txt**) containing UIDs that can be processed for use as input for the **data_share** utility:

```
sync_on_demand -f=report -uid_report=uids_list.txt
```

- To create both a report file (**uids_list.txt**) containing UIDs and a synchronization report file (**sync.txt**):

```
sync_on_demand -f=report -report_file=sync.txt
               -uid_report=uids_list.txt
```

- To create a report file (**uids_list.txt**) containing UIDs enclosed by brackets (**{ }**):

```
sync_on_demand -f=report -uid_report=uids_list.txt -sep="{ }"
```

- To create a report file showing the synchronization status of 4GD data between the local and target site:

```
sync_on_demand -f=report -type=4GD -recipe=Bridge  
-report_file=4gd_report.txt -site=CentralSite
```

t2c_create_installer

Use this utility to deliver the current Teamcenter **tc_customization_compare_report** and **t2c_schema_analyzer_report** migration utilities to a site running an earlier version of Teamcenter (the source site). Doing so is required for those utilities to run properly when migrating Teamcenter data using Multi-Site.

SYNTAX

```
t2c_create_installer
-f={create_installer | deploy}
-installer_location=output_dir
[-h]
```

ARGUMENTS

-f

Specifies the function to be performed; you must specify one of the following arguments:

create_installer

Bundles the files needed at the target site to run the command line utilities used for migrating Teamcenter data using Multi-Site. Use the **create_installer** argument on the target site (the site running the later version of Teamcenter).

deploy

Updates the source site so the command line utilities used for migrating Teamcenter data using Multi-Site can run properly. Use the **deploy** argument on the source site (the site running the earlier version of Teamcenter).

-installer_location

When used at the target site with **create_installer**, specifies the folder in which the files needed to deliver to the target site are saved. The files are saved in a file named *InstalledData.zip*.

When used with **deploy**, specifies the source site folder containing the unzipped *InstalledData* folder.

-h

Displays help for this utility.

ENVIRONMENT

This utility works on any Linux or Windows platforms with Perl installed and in the path.

RESTRICTIONS

None.

EXAMPLES

- Collect and compress the files to deliver to the source site. Run this command on the target site.

```
t2c_create_installer -f=create_installer -installer_location=c:\temp
```

This command creates the file *c:\temp\InstalledData.zip*.

- Update source site with the files needed to run the command line utilities. Run this command on the target site.

```
t2c_create_installer -f=deploy -installer_location=c:\temp
```

t2c_report_comparator

Compares the source and target site data .csv files created using the **t2c_report_extract** utility **-function=extract** argument. **t2c_report_comparator** identifies missing objects, extra objects, and attribute mismatches between source and target objects.

.csv files in the source site folder are compared with files of the same name in the target site folder. The source and target folder files must be created for the same sets of objects.

This utility is used as part of the process of migrating Teamcenter data to a newer version of Teamcenter.

SYNTAX

```
t2c_report_comparator  
-source=source_folder  
-target=target_folder  
-report_folder=report_folder  
[-h]
```

ARGUMENTS

-source

Specifies the folder containing the source site .csv file containing data extracted using the **t2c_report_extract** utility.

-target

Specifies the folder containing the target site .csv file containing data extracted using the **t2c_report_extract** utility.

-report_folder

Specifies the folder in which the report files are saved.

-help

Displays help for this utility.

ENVIRONMENT

This utility works on any Linux or Windows platforms with Perl installed and in the path.

RESTRICTIONS

Use this utility to compare the source and target site data .csv files created using the **t2c_report_extract** utility in preparation for using the Multi-Site **data_share** utility to migrate data between different Teamcenter releases.

t2c_report_extract

Use this utility to generate a migration status report for objects that have been registered using the **data_share** utility with its **-function=register_migration_data** argument. The utility can also be used to extract islands of data to .csv files from source and target sites for comparison using the **t2c_report_comparator** utility.

This utility is used as part of the process of migrating Teamcenter data to a newer version of Teamcenter.

SYNTAX

```
t2c_report_extract [-u=user-id {-p=password | -pf=password-file} -g=group]
-function={status_report -dir=output_directory[-class=class1,class2,...classN] [-non_migrated_list] |
extract -output_file=file_name [-filename=input_file_name
-classesfile=input_file_class] [-mapping_rule=rule_file -output_file=file_name] [-
optionset=option_set_name] [-session_options=option1:value1,option2:value2,...optionN:valueN]
[-override_options=option1:value1,option2:value2,...optionN:valueN] [-
adminclass=admin_class_name]
[-item_id=item_id_val] [-qry_name=saved_query_name -qry_attr=attr_name1 -qry_val=attr_val1[ -
qry_attr=attr_name2 -qry_val=attr_val2[ -qry_attr=attr_nameN -qry_val=attr_valN]]] [-since=time]}
[-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-function**status_report**

Saves the migration status report as an HTML page in the directory specified by **-dir**.

If the report identifies objects that have been updated on the source site since they were migrated, an `\OutOfSync` directory is created in the directory specified by the **-dir**. In that directory, `.txt` files are created for each type having out-of-sync objects.

If errors occurred during the migration, an `\Error` directory is created in the directory specified by the **-dir**. In that directory, `.txt` files are created for each type generating errors.

extract

Extracts the Teamcenter objects as an island of data, including related attributes, from the source and target sites. The data is saved in the file specified by **-output_file**. The source and target site data can then be compared using the **t2c_report_comparator** utility.

-dir

The directory in which migration status report files are saved. Used only with **-function=status_report**.

-class

The classes the utility is to report on. If missing Item is reported on, Separate multiple classes with commas. **-class** is used only with **-function=status_report**.

-non_migrated_list

By default, **-function=status_report** does not generate the (potentially long) list of objects not migrated. Specifying **-non_migrated_list** with **-function=status_report** generates a full list of objects not migrated.

-output_file

Specifies the path and name of the `.csv` file to which the island of data is saved. Required with **function=extract**

-filename

Specifies an input file containing the list of objects to report on. Each line in the file must contain only one object. If **-filename** is not provided, all objects are reported on. **-filename** is valid only with **-function=extract**.

-filename can contain the following lists:

- Objects of the same type

- Business object unique IDs
- Administrative object unique IDs
- Item objects in MFK key format. These objects must be listed using the form shown in the following example:

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

- Item Revision objects in MFK key format. These objects must be listed using the form shown in the following example:

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1,rev_id=A
```

The class of the objects in the file is specified with **-classoffile**.

-classoffile

The class of objects listed in the file specified by **-filename**. **-classoffile** is valid only with **-function=extract**. The following class values are supported:

Value	Description
Item, ItemRevision, Dataset, Form, Folder	The file contains objects of one of these types.
uidfile	The file contains business object unique IDs.
adminuidfile	The file contains administrative object unique IDs.
itemkeyfile	The file contains Item objects in MFK key format.
itemrevisionkeyfile	The file contains Item Revision objects in MFK key format.

-mapping_rule

For scenarios where data on the source site must be transformed (mapped) to different objects and attributes on the target site, **-mapping_rule** specifies the mapping rule .csv file used when transforming the data. Specifying **-mapping_rule** with **-function=extract** results in two .csv files: *file_name.csv* (the specified island of data at the source site) and *file_name_mapped.csv* (the transformed island of data).

You can run **t2c_report_extract** at the target site and then use the **t2c_report_comparator** utility to confirm the source objects and attributes are mapped correctly to the existing target site objects and attributes.

For example,

1. At the source site, use Advanced Multi-Schema Exchanger to create a mapping rules file for migration. For example, *mapping_file.csv*.

2. On the source site, run:

```
t2c_report_extract -f=extract -item_id=xxx
-output_file=d:\source.csv -mapping_rule=mapping_file.csv
```

t2c_report_extract generates *source.csv* (the source site data) and *source_mapped.csv* (the mapped data).

3. On the target site, run:

```
t2c_report_extract -f=extract -item_id=xxx
-output_file=d:\target.csv
```

t2c_report_extract generates a folder containing *target.csv*, the island of data as it exists on the target site.

4. Compare the source site mapped data to the actual target site data using **t2c_report_comparator**. For example,

```
t2c_report_comparator -source=d:\source_mapped_folder
-target=d:\target_folder
```

Where:

-source

Contains *source.csv* and *source_mapped.csv*.

-target

Contains *target.csv*.

t2c_report_comparator generates a report verifying that the source objects and attributes are mapped correctly to the target objects and attributes.

-optionset

Specifies the option set to use when extracting data. **-optionset** is valid only with **-function=extract**.

-item_id

Specifies one or more **item_id** values to report on. Wildcard characters (*) are supported.

-adminclass

Reports on all instances of a specified administrative class. Used when the file specified by **-filename** contains administrative object unique IDs and **-classoffile** is set to **adminuidfile**.

-qry_name

Runs a saved Query Builder query that defines the list of objects to process. One or more **-qry_attr** and **-qry_val** pairs specify the attribute value pairs the objects must match to be processed. The pairs are separated by spaces. Enclose attribute names containing spaces with double quotes.

-session_options

Specifies generic session options using colon separated name-value pairs. Name-value pairs can contain multiple values paired with a name. Options are delimited by commas. Multiple values in pairs are delimited by semicolons as in the following example.

```
-session_options=opt1:value1,opt2:value2,opt3:value3a;value3b
```

-override_options

Used with **-optionset** to override options defined in the named option set. Specify options to override with colon-separated name-value pairs. Delimit multiple name-value pairs with commas as in the following example.

```
-override_options=opt1:value1,opt2:value2,opt3:value3
```

-since

Reports on only those objects modified or out of sync since the specified date and time. Specify the date and time using the format **YYYY-MM-DD:HH:mm** format. For example, **2023-06-21:00:30** represents June 21, 2023, 12:30 am. **HH:mm** are optional, and default to **00:00**.

-h

Displays help for this utility.

RESTRICTIONS

Use this utility in preparation for using the Multi-Site **data_share** utility to migrate data between different Teamcenter releases.

EXAMPLES

- Generate migration status report for the class Item and ItemRevision

```
t2c_report_extract -f=status_report -dir=d:\xyz_01
-class=Item,ItemRevision
```

- Extract the specific item data to the CSV file.

```
t2c_report_extract -f=extract -output_file=d:\xyz.csv -item_id=xyz
```

t2c_schema_analyzer_report

The **Multi-Site data_share utility** supports bulk migration of data between different Teamcenter releases, for example when moving from an earlier on-premises Teamcenter release to a later cloud-based release. You can run the **t2c_schema_analyzer_report** utility before the migration to analyze the data model templates of the two installations to identify any changes needed for the source site data to be compliant with the target site's data model. Addressing those changes enables you to avoid the need to upgrade the source site to the same version as the target site before the migration.

You can run **t2c_schema_analyzer_report** from either the source or target site. Both sites must be running Teamcenter 11.6 or later with TC XML-based Multi-Site installed. See Migrating Multi-Site Teamcenter data to a newer version of Multi-Site for more information.

Running **t2c_schema_analyzer_report** produces .csv and .xml reports showing the business object and attribute changes needed for the data to be compatible with the data model of the later version of Teamcenter. The .xml report summarizes the changes. The .csv report contains a more detailed breakdown of the needed changes. See Understanding the results for more information on the schema analyzer reports.

Running **t2c_schema_analyzer_report** also produces a script that programmatically addresses certain data compatibility issues between the Teamcenter versions. Run this script after migrating your data. Other data changes may require you to modify the data model at the source site.

SYNTAX

```
t2c_schema_analyzer_report
-version=from_version
-target_version=from_version
-report_folder=report_path
{-source=source_template_path | -target=target_template_path}
[-db_profiling_file=profiling_file]
[-help]
```

ARGUMENTS

-version

Specifies the version of Teamcenter on the source site.

Use the form **TC*major.minor***. For example, **TC12.3** and **TC 13.1**. You can further include patch version numbers if the site has been patched. For example, **TC12.3.0.1**.

When running Teamcenter, select **Help > About** to verify the version of Teamcenter.

-target_version

Specifies the version of Teamcenter on the target site. Use the same form as described for **-version**.

-report_folder

Specifies the folder in which the report and batch files are saved.

-source

Required when you are running the utility on the target site. Specifies the directory containing the source site's template files. These files are stored in the source site's *TC_DATA\model* directory.

-target

Required when you are running the utility on the source site. Specifies the directory containing the target site's template files. These files are stored in the target site's *TC_DATA\model* directory.

-db_profiling_file

Specifies the data profile .csv file containing the object count for each type of business object in the database. Optionally create this file using the steps in *Create a profile of the source site's data* in Performing the analysis.

-help

Displays help for this utility.

ENVIRONMENT

This utility works on any Linux or Windows platforms with Perl installed and in the path.

RESTRICTIONS

Use this utility in preparation for using the Multi-Site **data_share** utility to migrate data between different Teamcenter releases.

EXAMPLES

- Run from the target site to analyze a local copy of the source site release 12.1 templates against the current site's templates in its *TC_DATA_MODEL* folder.

```
t2c_schema_analyzer_report.bat -version=12.1
-report_folder=c:\temp\report
-source=c:\temp\source_templates
-db_profiling_file=c:\temp\tc121object_20230621.csv
```

If a value for **-db_profiling_file** is not provided, all business objects in the source schema (including unused business objects) are analyzed.

- Run from the source site to analyze the source site's current templates against the target site's templates available using a mapped drive.

```
t2c_schema_analyzer_report -version=12.1 -report_folder=c:\temp\report
-db_profiling_file=c:\temp\tc121object_20230621.csv
-target=m:\tc14\TD\model
```

tc_attribute_bulk_update

Queries for specified properties and their current values, and then exports specified updated values into a TC XML file and then imports the TC XML file containing the updated values into the database.

You can use this utility to update the properties of numerous objects in one pass. The update process allows you to update one or more attributes of an object. When a specified object contains multiple attributes, only the specified attributes are exported and updated.

For complex update operations, you can create an input XML file containing instructions on which objects to update and the new values to apply. The file is constructed as a series of **UpdateSets** entries. Each entry must contain **type**, **where**, and **update** components.

Before performing lengthy update operations, you can use the utility to determine the number of objects affected by a specified update operation.

SYNTAX

```
tc_attribute_bulk_update -u=user-id {-p=password | -pf=password-file} -g=group
{-inputfile=path-to-input-XML-file -type=type-name
-update_prop=property-name -update_value=property-value
-cond_prop=property-name -cond_value=value-name}
[-cond_operator=operator]
[performSchemaValidation]
[-queryonly]
[-outdir=output-XML-file-directory]
[-batchsize=number-of-objects-to-update]
[-untransformed]
[-uidfile]
[-import]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputfile

Specifies the full path to the input XML file containing instructions on which objects to update and the new values to apply. The file is constructed as a series of **UpdateSets** entries. Each entry must contain **type**, **where**, and **update** components. Use the input file for complex updating instances. (For simpler instances, you can use the **-cond_prop**, **-cond_value**, and **-cond_operator** arguments.) You must use either this argument, or the **-type**, **-update_prop** and **-update_value** arguments.

Following is an example input file:

```
<?xml version=1.0" encoding="UTF-8"?>
<BulkUpdate>
  <UpdateSets>
    <UpdateSet>
      <type name="ImanVolume" />
      <where>
        <cond_prop attrName=volume_name" attrValue="Volume1" cond_operator="=" />
      </where>
      <update>
        <update_prop attrName="node_name" attrValue="s0928933" />
        <update_prop attrName="machine_type" attrValue="0" />
        <update_prop attrName="shadow_unix_path_name" attrValue="" />
        <update_prop attrName="shadow_vms_path_name" attrValue="" />
        <update_prop attrName="shadow_wnt_path_name" attrValue="" />
        <update_prop attrName="wnt_path_name"
attrValue="E:\workdir\install\tc11mad1\TV" />
        <update_prop attrName="unix_path_name" attrValue="" />
        <update_prop attrName="vms_path_name" attrValue="" />
      </update>
    </UpdateSet>

    <UpdateSet>
      <type name="ImanVolume" />
      <where>
        <cond_prop attrName=volume_name" attrValue="Volume2" cond_operator="=" />
      </where>
      <update>
        <update_prop attrName="node_name" attrValue="s0987324" />
        <update_prop attrName="machine_type" attrValue="2" />
        <update_prop attrName="shadow_unix_path_name" attrValue="" />

```



```

        <update_prop attrName="shadow_vms_path_name" attrValue="" />
        <update_prop attrName="shadow_wnt_path_name" attrValue="" />
        <update_prop attrName="wnt_path_name" attrValue="" />
        <update_prop attrName="unix_path_name" attrValue="/users/jpip/volumes/
tv2" />
        <update_prop attrName="vms_path_name" attrValue="" />
    </update>
</UpdateSet>
</UpdateSets>
</BulkUpdate>

```

-type

Specifies the type of objects to be updated, for example, item, dataset, **ImanVolume**, and so on. This argument accepts a single value, which must be a valid Teamcenter object. Use multiple instances of this argument to specify multiple object types. You must either use this argument in conjunction with the **-cond_prop**, **-cond_value**, **-update_prop**, and **-update_value** arguments, or use the **-inputfile** argument.

-update_prop

Specifies the internal name of the property to be updated (as opposed to the display name), for example, **object_desc**, **char VLA**, and so on. This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object, for example:

```
-update_prop=object_name, object_desc
```

You can use multiple instances of this argument. Each instance of this argument should be paired with the **-update_value** argument. You must either use this argument in conjunction with the **-cond_prop**, **-cond_value**, **-type**, and **-update_value** arguments, or use the **-inputfile** argument.

-update_value

Specifies the new value for the property specified by the **-update_prop** argument. This argument accepts multiple values in a comma-separated list, for example:

```
-update_value=folder, "Home folder"
```

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-update_prop** argument. You must either use this argument in conjunction with the **-cond_prop**, **-cond_value**, **-type**, and **-update_prop** arguments, or use the **-inputfile** argument.

-cond_prop

Specifies the name of the property to be queried for and updated during export. This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object. For example:

```
-cond_prop=object_name, last_mod_date
```

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-cond_value** argument. You must either use this argument in conjunction with the **-cond_value**, **-type**, **-update_prop**, and **-update_value** arguments, or use the **-inputfile** argument.

-cond_value

Specifies the new value for the property specified by the **-cond_prop** argument. This argument accepts multiple values in a comma-separated list. Each value must be a valid property on a Teamcenter object. For example:

```
-cond_value=TextData_es_ES, "01-DEC-2011 00:00"
```

You can use multiple instances of this argument. Each instance of this argument must be paired with the **-cond_value** argument. You must either use this argument in conjunction with the **-cond_prop**, **-type**, **-update_prop**, and **-update_value** arguments, or use the **-inputfile** argument.

-cond_operator

Specifies the operator to be used with the condition arguments. Valid operations are: equal, not equal, greater than, greater than and equal, less than, less than and equal. When setting the value for this argument in the command line, the format for these operations are: **EQ**, **NE**, **GT**, **GE**, **LT**, **LE**. When setting the value for this argument in the input file, you can use this format or the following characters: **=**, **!=**, **>**, **>=**, **<**, **<=**. This argument is optional and is only valid when used with, and placed after, the **-cond_prop** and **-cond_value** arguments, for example:

```
-cond_prop=object_name -cond_value="My obj #1"
-cond_prop="last_mod_date" -cond_value="01-DEC-2011 00:00"
-cond_operator="LE" -cond_prop="last_mod_date"
-cond_value="01-DEC-2010 00:00" -cond_operator="GE"
```

-performSchemaValidation

Enables schema validation of the XML input file.

-queryonly

Outputs the number of target objects affected by the specified update parameters to a log file. If the number of objects is less than 100, the object UIDs are included in the log file. When you specify this switch, the utility does not perform the update, it merely reports the number of affected objects. Use this switch in conjunction with either the input file or the condition and update arguments to determine how many objects are affected by specified update operation.

You can use the resulting information to determine batch size, and to estimate the duration of the update operation.

-outdir

Specifies the directory to which the TC XML file and import log is generated. The file name is automatically generated and imported into the database using the following naming convention:

```
BulkAttrOut_1.xml, BulkAttrOut_2.xml, BulkAttrOut_3.xml
```

This argument is optional. If not specified, the current directory is used.

-batchsize

Specifies the number of objects to update per TC XML file. This argument is optional. The default value is **800**.

-untransformed

Exports the original values of the specified attributes. This argument is optional.

-uidfile

Specifies the full path to the input XML file containing a list of object UIDs. This argument is optional. If specified, it suppresses the function of the **-cond_prop** argument.

-import

Imports the TC XML file generated into the database for updating the property values given in the input file. This argument is optional.

-h

Displays help for this utility.

RESTRICTIONS

You must be a privileged user to run this utility. You must use either the **-inputfile** argument or the **-type**, **-update_prop**, **-update_value**, **-cond_prop**, and **-cond_value** arguments.

Note:

The condition arguments are the equivalent of the **where** clause in an SQL phrase. Running this utility without specifying the **-cond_prop** and **-cond_value** arguments is technically possible but impractical.

EXAMPLES

- To update the attributes of certain properties as specified in the *propFile.xml* file, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -p=admin-password -g=dba
-inputfile=d:\propFile.xml -log=logFile.txt
```

- To update **prop3** to **value3** and **prop4** to **value4** for **type1** objects, when **prop1** equals **value1** and **prop2** is greater than **value2**, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -pf=d:\password.txt -g=dba -type=type1
-cond_prop=prop1 -cond_value="value1" -cond_prop=prop2 -cond_value="value2"
-cond_operator="GT" -update_prop=prop3 -update_value="value3" -update_prop=prop4
-update_value="value4"
```

- To update **prop3** to **value3** and **prop4** to **value4** for **type1** objects, when **prop1** equals **value1** and **prop2** is greater than **value2** in batches of **400**, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -pf=d:\password.txt -g=dba -type=type1
-cond_prop=prop1 -cond_value="value1" -cond_prop=prop2 -cond_value="value2"
```

```
-cond_operator="GT" -update_prop=prop3 -update_value="value3"
-update_prop=prop4 -update_value="value4" -batchsize=400
```

- To update an object description to **red Desc** and the object name to **This is a new object name** for all items where the **object_name** property is **my object** and the **str_attr** property is **red,white,yellow**, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -p=admin-password -g=dba -type=Item
-update_prop=object_desc -update_value="red Desc" -update_prop=object_name
-update_value="This is a new obj name"
-cond_prop=object_name -cond_value="my obj" -cond_value="str attr"
-cond_value="red,white,yellow"
```

- To update an **object description** to **blue Desc** and the **int_VLA** to **10,3,0,99** for all datasets where the **object_name** property is **TextData_es_ES** and the **last_mod_date** property is **01-DEC-2011 00:00** or greater, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -p=admin-password -g=dba -type=Dataset
-update_prop=object_desc -update_value="blue Desc"
-update_prop="int_VLA" -update_value="10,3,0,99" -cond_prop=object_name
-cond_value="TextData_es_ES" -cond_prop="last_mod_date" -cond_value=
"01-DEC-2011 00:00" -cond_operator="GE"
```

- To update the **char_VLA** to **w,y,d,k,a** for all items where the **owning_user** property is **AsL5bfuW4ghudD**, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -p=admin-password -g=dba -type=Item
-update_prop="char_VLA" -update_value="w,y,d,k,a" -cond_prop=owning_user
-cond_value="AsL5bfuW4ghudD"
```

- To update an object description to **blue Desc**, and to update the **int_VLA** property to **10,3,0,99** for all datasets where the **object_name** property is **TextData_es_ES** and the **last_mod_date** property is **01-DEC-2011 00:00** or greater, and to place the changes directly into the database by importing the TC XML file in single step, enter the following command on a single line:

```
tc_attribute_bulk_update -u=admin-user -p=admin-password -g=dba
-type=Dataset -update_prop=object_desc -update_value="blue Desc"
-update_prop="int_VLA"
-update_value="10,3,0,99" -cond_prop=object_name -cond_value="TextData_es_ES"
-cond_prop="last_mod_date" -cond_value="01-DEC-2011 00:00" -cond_operator="GE"
-import
```

tc_customization_compare_report

Analyzes Teamcenter sites and reports BMIDE customizations. This analysis is particularly useful when moving from an earlier on-premises Teamcenter release to a later cloud-based release. Run this utility using the **-f=create_package** argument on two sites to generate packages of data model and template files. Then, use this utility using the **-f=generate_report** argument to compare those packages and generate reports of differences as described in Identify and resolve BMIDE customization differences.

SYNTAX

```
tc_customization_compare_report [-u=user-id {-p=password | -pf=password-file} -g=group]
-f={create_package -out=output_folder |
generate_report
-type={bmide | adminData | all}
-source=model_path
-target=model_dir
-sourceVersion=source_ver
-targetVersion=target_ver
[-skip_added]
[-skip_admin]}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Required to use **-f** with a value of **generate_report**. This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see

For more information about managing password files, see Manage password files.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

When using **-f** with a value of **generate_report**, this is a user who is a member of a group with DBA privileges.

If used without a value, the user's default group is assumed.

-f

Specifies the function to be performed; you must specify one of the following arguments:

create_package

Generates a package (folder) containing data model and template files used at the current site. Run the utility at both the source and target sites to generate packages for comparison. Specify a value for **-out** when using **create_package**.

generate_report

Compares two packages of data model and template files and generates reports of their differences.

-out

Specifies the folder in which the generated package (folder) of model and template files is stored. Use with **-f=create_package**.

-type

Specifies the report type to generate. A value of **bmide** generates a report comparing the data models and templates. A value of **adminData** generates a report comparing the sites' administration data. A value of **all** (or **bmide,adminData**) generates reports for both types of data.

-source

Specifies the directory containing the folder containing the package (folder) of files generated from the source site. (This is the earlier version or on-premises site). Use with **-f=generate_report**.

-target

Specifies the directory containing the folder containing the package (folder) of files generated from the target site. (This is the later version or cloud-based site). Use with **-f=generate_report**.

-sourceVersion

Specifies the version of Teamcenter on the source site. Use with **-f=generate_report**.

Use the form **TCrelease**. For example, **TC2312** and **TC2406**.

When running Teamcenter, select **Help > About** to verify the version of Teamcenter.

-targetVersion

Specifies the version of Teamcenter on the target site. Use the same form as described for **-sourceVersion**.

-report_folder

Specifies the folder in which the generated report files are saved. Use with **-f=generate_report**.

-skip_added

Excludes added entries when generating reports. Use with **-f=generate_report**.

-skip_admin

Excludes administration data when generating reports. Use with **-f=generate_report**.

-h

Displays help for this utility.

ENVIRONMENT

This utility works on any Linux or Windows platforms with Perl installed and in the path.

RESTRICTIONS

Use this utility in preparation for using the Multi-Site **data_share** utility to migrate data between different Teamcenter releases.

EXAMPLES

- Run at the current site to generate a package to compare data model and template files.

```
tc_customization_compare_report -f=create_package -type=bmide
-out=c:\temp\TC2312
```

- Generate data model and template file comparison reports.

```
tc_customization_compare_report -u=admin-username -p=password
-f=generate_report -type=bmide -target=c:\temp\TC2406
-source=c:\temp\TC2312 -report_folder=c:\temp\TcCompare
-sourceVersion=TC2312 -targetVersion=TC2406
```

The comparison report is saved to the `c:\temp\TcCompare` directory.

tcxml_clean_sync_tables

Provides options for optimizing TC XML data in synchronization tables.

SYNTAX

```
tcxml_clean_sync_tables -u=user-name {-p=password | -pf=password-file} [-g=group]
{-quiet}
{-delete_subscription_table [-app_ids=application-ids]}
{-update_subscription_table [-app_id=app_ids] [-lpd=lpd]}
{-delete_scratch_table {-null_lsd} {-stale_data} {-classes=class-names} {-truncate}}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. The file must be a single-line ASCII file containing the password in clear text. Teamcenter Environment Manager prompts you for a password and creates the password file during installation.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-quiet

Runs the utility with no confirmation prompts when deleting or updating the synchronization tables.

-delete_subscription_table

Removes data from SUBSCRIPTION_TABLE. Used with the **-app_ids** argument.

-app_ids

Specifies application IDs to be removed from SUBSCRIPTION_TABLE. Multiple IDs are delimited with commas. Used with the **-delete_subscription_table** argument.

-update_subscription_table

Updates the last processed date (lpd) in SUBSCRIPTION_TABLE for the ID specified by **-app_id**. Used with the **-app_id** and **-lpd** arguments.

-app_id

Specifies the ID of the lpd in SUBSCRIPTION_TABLE to be updated by **-update_subscription_table**.

-lpd

Specifies the value **-update_subscription_table** uses to update the last processed date in SUBSCRIPTION_TABLE. The value has the format `"DD-MMM-YYYY HH:MM:SS"` (double quotes included).

- *DD* specifies the day (01-31)
- *MMM* specifies the month (jan-dec). Letter case is not significant.
- *YYYY* specifies the year (0001-9999)
- *HH* specifies the hour (00-23)
- *MM* specifies the minute (00-59)
- *SS* specifies the second (00-59)

-delete_subscription_table

Removes data from SCRATCH_TABLE. Used with the **-null_lsd**, **-stale_data**, **-classes**, and **-truncate** arguments.

-null_lsd

Removes the NULL lsd records from SCRATCH_TABLE. Used with the **-delete_subscription_table** argument.

-stale_data

Removes already-synchronized data from SCRATCH_TABLE. Used with the **-delete_subscription_table** argument.

-classes

Specifies the names of data classes to remove from SCRATCH_TABLE. Multiple class names are delimited with commas. Used with the **-delete_subscription_table** argument.

-truncate

Removes all data from (truncates) SCRATCH_TABLE. Used with the **-delete_subscription_table** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter* Log files produced by Teamcenter.

RESTRICTIONS

None.

EXAMPLES

Removes two application IDs from SUBSCRIPTION_TABLE:

```
tcxml_clean_synch_tables -u=Tc-admin-user -p=password -g=group
-delete_subscription_table -apps=app01,app_02
```

Updates the last processed date for a specific application ID in SUBSCRIPTION_TABLE:

```
tcxml_clean_synch_tables -u=Tc-admin-user -p=password -g=group
-update_subscription_table -app_id=app01
-lpd="31-Aug-2018 13:20:00"
```

Removes NULL LSD data from SCRATCH_TABLE:

```
tcxml_clean_synch_tables -u=Tc-admin-user -p=password -g=group
-delete_scratch_table -null_lsd
```

Removes a list of data classes from SCRATCH_TABLE:

```
tcxml_clean_synch_tables -u=Tc-admin-user -p=password -g=group -quiet
-delete_scratch_table -classes=class01,class02,class03
```

Removes all the data from SCRATCH_TABLE:

```
tcxml_clean_synch_tables -u=Tc-admin-user -p=password -g=group
-delete_scratch_table -truncate
```

tcxml_export

Exports objects from Teamcenter in the TC XML format. If there are files for export, the utility creates a File Management System (FMS) read file ticket and saves it in the output XML file for each file.

Note:

Do not use the **tcxml_export** utility for exporting administration data. For exporting administration data, use the **admin_data_export** utility.

SYNTAX

```
tcxml_export [-u=user-id [-p=password | -pf=password-file] [-g=group] ]
-file=output-xml-file {-item=item-id [-rev=revision-id] | -folder=folder-name |
-class=POM-classname | -uid=uid-of-object |
-item_key=attr-name1=value, attr-name2=value, ... |
-inputfile=file-name | -inputuidfile=file-name}
[-transfermode=transfer-mode-name | -optionset=transfer-option-set-name]
[-targetsites=list-of-target-site-ids]
[-transferownership]
[-sync]
[-incrementalChangeDelta]
[-force_reexport]
[-reason=reason-for-export]
[-revrule =revision-rule]
[-bomlevel =desired-bom-level]
[-svrule=saved-variant-rule-name [:rev-id:rule-id]]
[-processUnconfiguredByOccEff]
[-processSuppressedOcc]
[-processUnconfiguredVariants]
[-processUnconfiguredChanges]
[-baseline_id]
[-baseline_rev]
[-generateBOMIndex]
[-fromBOMIndex]
[-xsl=xsl-file-name]
[-session_options=option-1:value-1,option-2:value-2,...option-n:value-n]
[-session_options_file=options-file-name]
[-requiredLang=locale-code-1, locale-code-2, ..., locale-code-n]
[-allowedLang=locale-code-1, locale-code-2, ..., locale-code-n]
[-briefcase]
[-dryrun [=validateXMLBeforeXslt]]
[-validate [=validateXMLBeforeXslt]]
[-low-level {-inputfile=file-with-item-ids | -inputuidfile=file-with-uids}
[-bulk_extract]
[-input_criteria=class-name{attribute1=value1,attribute2=value2,...attributen=valuen}]
[-force_retraverse]]
```

[-verbose]
[-h]

ARGUMENTS

-u

Specifies the user ID.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the output file name. The value can be either an absolute path (full path name) or a relative path name. By default, the output file contains XML. When the **-dryrun** or **-validate** option is specified, the file contains an HTML report of any warnings or errors in the TC XML. When using the **-dryrun** or **-validate** option, append a **.html** extension to the file name.

-item

Specifies the **item_id** values of one or more items to be exported. Multiple values are separated by colons.

-rev

Specifies the revision ID of the item to be exported. If this argument is not specified, the configured revision (either specified or default) is exported for the item.

-folder

Specifies the folder containing objects to be exported.

-class

Specifies a class name. Instances of this POM class are exported. The following workspace object names are valid:

- **Item**
- **ItemRevision**
- **Folder**
- **Dataset**
- **Alias**
- **ImanFile**
- **ImanRelation**
- **ReleaseStatus**
- **IdContext**
- **Identifier**
- **PSBOMView**
- **PSBOMViewRevision**
- **TransferMode**
- **TransferOptionSet**

-uid

Specifies the UID of an object (one object only).

-item_key

Specifies a comma-delimited list of attributes used to identify the object to be exported.

-item_key supports multifield keys using the following form:

```
-item_key=attribute_name_1=value_1, attribute_name_2=value_2, ...
```

-inputfile

Specifies the name of a text file containing the item IDs of objects you want to export. **-inputfile** also supports multifield keys. The file must have separate lines for the item attributes you want to use to identify the objects, for example:

```
item_id=export_001
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

-inputuidfile

Specifies the name of a text file containing the UIDs of objects you want to export. The file must have separate lines containing the object UID. For lightweight object UIDs, provide the UID and the object class separated by a colon.

-transfermode

Specifies the transfer mode name used to export the objects. If this argument is not specified, the utility uses a default transfer mode. See restriction 2.

-optionset

Specifies the transfer option set name used to export the objects. Mutually exclusive with the **-transfermode** argument. If you specify both, the command does not fail. However, the transfer

mode indicated by this **-optionset** argument takes precedence over the transfer mode specified by the **-transfermode** argument. See restriction 3.

-targetsites

Specifies a comma-delimited list of destination site IDs. If used with the **-transferownership** argument, **-targetsites** must contain only one site ID.

-transferownership

Indicates that this export transfers the ownership of exported objects to the specified target site. You must specify only one site ID in the **-targetsites** argument if you use this argument.

-sync

Indicates that this export is for data synchronization.

-incrementalChangeDelta

Exports modified objects tracked by configured incremental change as a partial structure export.

Because incremental change data is configured data, you must specify a configured transfer option set for the **-optionset** argument. The **-processUnconfiguredChanges** and **-processSuppressedOcc** arguments are ignored if they are included.

-force_reexport

Repeats the most recent configured export specified by the values of the following arguments: **-item**, **-targetsites**, **-revrule**, and **-svrule**.

Configured exports of high level and low level TC XML to managed or unmanaged sites are supported. Full and partial BOM exports are also supported.

Objects do not need to be marked for ownership transfer. The same ownership transfer specifications are used as in the previous export. Objects with their ownership transferred in the previous export are also included in this export.

If the previous export meeting this criteria was a full export, the full export is repeated, including changes since the previous full export. If there is no previous export, a full export is created.

-reason

Specifies the reason for this export.

-revrule

Specifies the revision rule used to configure the exported BOM with the specified item as the top line.

-bomlevel

Specifies the level in the BOM.

-svrule

Specifies a saved variant rule to use when configuring the exported BOM. Optionally, specify a particular revision of a variant rule object by providing its revision ID and rule ID separated by

colons. Variant rule revision use is supported for only **Cfg0VariantCriteria**, a revisable subclass of the **VariantRule** object.

-processUnconfiguredByOccEff

Exports **BOMLine** objects that are not configured for occurrence effectivity.

-processSuppressedOcc

Exports suppressed **BOMLine** objects.

-processUnconfiguredVariants

Exports **BOMLine** objects that are not selected by **BOMLine** object's variant conditions.

-processUnconfiguredChanges

Exports **BOMLine** objects configured out of the BOM by incremental change.

-baseline_id

Specifies the baseline ID for exporting a configured minor revision.

-baseline_rev

Specifies the baseline revision ID for exporting a configured minor revision.

-generateBOMIndex

Saves **BOMLine** data to persistent tables. The TC XML data is not serialized.

-fromBOMIndex

Exports the **BOMLine** data directly from the persistent cache without configuring and expanding the BOM.

-xsl

Specifies the output XSL file to apply to the TC XML file after export.

-session_options

Specifies a comma-separated list of option name-value pairs. Names and values are delimited by colons, for example:

`ContinueOnError:TRUE,GenerateReport:TRUE`

-session_options_file

Specifies an ASCII file containing session option settings using option name-value pairs. Use this option instead of the **-session_options** option when you are reusing long lists of session name-value pairs and when values contain system-reserved special characters.

Place one name-value pair on each line of the file as follows:

```
option_1:value_1
option_2:value_2
.
.
```

`option_n:value_n`

Set an option only once in the file.

-requiredLang

Specifies a list of comma separated locale values. This list is used to ensure that localized attributes in the exported data have at least one representation that can be used as the attribute primary language at the importing site. It also defines a priority order for the exporter to determine the attribute primary language. The valid locale values must match the Java locale naming convention that consists of two groups of two-character identifiers separated by an underscore character (`_`) for a particular combination of language and region. For example, **zh_CN** represents Simplified Chinese in China and **en_US** represents English in the United States.

-allowedLang

Specifies a list of comma separated locale values. This list is used to get additional representations for localized attributes in the exported data for use at the importing site. The valid locale values must match the Java locale naming convention that consists of two groups of two-character identifiers separated by an underscore character (`_`) for a particular combination of language and region. For example, **zh_CN** represents Simplified Chinese in China and **en_US** represents English in the United States.

-briefcase

Specifies the output be formatted as a briefcase file. The file is specified with the **-file** option. **-briefcase** is only supported when exporting to managed sites. Sites must be specified using **-targetsites**.

-dryrun

Specifies a simulated low-level TC XML export be run and the exported TC XML be validated. Warnings and errors in the TC XML are reported in the file specified by **-file**. If **-dryrun=validateXMLBeforeXslt** is specified, the internal (pre-export) TC XML is validated instead of the exported TC XML. A briefcase file not created in either case. **-dryrun** cannot be used with the **-validate** option.

-validate

Specifies that the exported TC XML be validated. If no warnings or errors are detected in the TC XML, the briefcase file is created. If warnings or errors are detected in the TC XML, they are reported in the file specified by **-file** (with a `.html` extension) and no briefcase file is created. If **-validate=validateXMLBeforeXslt** is specified, the internal (pre-export) TC XML is validated instead of the exported TC XML. **-validate** cannot be used with the **-dryrun** option.

-h

Displays help for this utility.

FAST EXPORT ARGUMENTS

The following arguments support low-level fast export functions used for site consolidation activities. The use of these arguments requires the **SITCONS_AUTH_KEY** environment variable be set to a valid license key. Open a support case on Support Center to get this license key.

Export files created using the low-level export do not contain global stable identifier (**GSID**) attributes.

-low_level

Performs fast export using POM-level APIs as a DBA user. You must specify this argument to use any of the fast export arguments.

In addition to the other fast export arguments, you can use any of the following standard **tcxml_export** arguments when you specify the **-low_level** argument.

- folder**
- file**
- targetsites**
- transfermode**
- optionset**
- reason**

Note:

The **-requiredLang**, **-allowedLang**, and **-transferownership** arguments are ignored if you supply them with the **-low_level** argument.

-bulk_extract

Extracts product data into a briefcase (.bcz) file that contains low-level TC XML and associated physical files used to bulk load the data into a test environment. This briefcase file is explicitly for a test environment and cannot be used for exchanging data with suppliers.

You can supply the **-optionset** argument if the transfer options set has its **opt_bulk_extract_bcz** option set to **TRUE**. If you do not specify the **-optionset** argument, the utility uses the **UnconfiguredBulkExtractDefault** transfer option set.

The **-force_retraverse** argument is set with the **-bulk_extract** argument. The following arguments are ignored if you supply them with the **-bulk_extract** argument:

- allowedLang**
- incrementalChangeDelta**
- requiredLang**
- sync**
- targetsites**
- transferownership**

If you do not have read and import privileges on an object when using the **-bulk_extract** argument, the object is exported as a stub in the TC XML file and the object's status in the export log is listed as `STUB_INSUFFICIENT_PRIVILEGE`. For example:

```
id14 [wuQtjENZAAgcRA] of type [Item ] - STUB_INSUFFICIENT_PRIVILEGE
```

-input_criteria

Identifies the criteria for specifying root objects for export. The class name and attribute values are used to search for the object or objects you want exported. Only single value attributes, including attributes from the parent classes, are supported. Subclasses are not included. Only the AND condition is allowed between different attributes. You can specify an attribute only once. Wildcard characters are supported as defined in the **TC_pattern_match_style** preference.

For example, to export all items with a 6 character item ID starting with **0000** and the object name starting with **Top**:

```
tcxml_export -input_criteria=Item{item_id=0000??,object_Name=Top*}  
-bulk_extract -file=abc_top.bcz
```

No other special character operators are supported. For example, the following characters are not supported and cannot be used in the class name, attribute name, or attribute value:

```
{ } = ,
```

The attribute value for a date range must be in the following format:

```
attribute-name="start-date to end-date"
```

For example, to specify objects created from 20 March to 1 April:

```
creation_date="20-Mar-2014 04:00 to 1-Apr-2014 04:00"
```

For an object reference type attribute, use the UID as the attribute value. This is true for any TC XML export.

-inputfile

Specifies a file that contains a list of item IDs indicating items to export using fast low-level export. You must include either this argument or the **-inputuidfile** argument when using the **-low_level** argument.

If your Teamcenter environment uses multifold key identifiers, you must specify the multifold key values for the **item_id** attribute as a list of comma-separated values in the input file, for example:

```
item_id=M2Item1_001,object_name=M2_Item_name1,object_type=M2Item1
```

The input file may contain both multifold key and standard item ID values, for example:

```
Item_id=Ace1
Item_id=lor1, object_name=fixedPl, object_type=type
Item_id=lor2, object_descr=acmetool, object_type=type
```

-inputuidfile

Specifies a file that contains a list of UIDs indicating items to export using fast low-level export. You must include either this argument or the **-inputfile** argument when using the **-low_level** argument.

-force_retraverse

Forces retraversal of previously replicated or exported objects during fast low-level export.

-verbose

Displays additional details during progress updates.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. Not all PLM data is supported. For a list of objects that are supported, see *Teamcenter Data Exchange*.
2. If you specify the **TransferMode** or **TransferOptionSet** object as the **-class** argument value, you are not required to specify the **-transfermode** or **-optionset** arguments. A predefined transfer mode is used for exporting these objects. If these arguments are specified they are ignored.
3. To export related dataset files, you must specify the **-transfermode** argument. The arguments value must be set to an option set containing closure rules that traverse dataset files related to the primary object. The **TIEExportDefaultTM** transfer mode contains a standard option set that can be used for this purpose.

EXAMPLES

- Select an item and export the item and its attachments using the default export transfer mode. The output XML file, **exportitem.xml**, is created in the directory where this command is executed.

```
tcxml_export -item=item_ID -file=exportitem.xml
```

- Select an item revision and export its attachments using the default export transfer mode. The output XML file, **itemrev.xml**, is created in the directory where this command is executed.

```
tcxml_export -item=item_ID -rev=item_rev -file=itemrev.xml
```

- Export the contents of the **exportObjects** folder using the default export transfer mode. If objects in the folder are supported objects, they are also exported. The output XML file, **folder.xml**, is created in the directory where this command is executed.

```
tcxml_export -folder=exportObjects -file=folder.xml
```

- Export item **000001** using the **TIEUnconfiguredExportDefault** transfer mode.

```
tcxml_export -item=000001 -file=exportitem.xml  
-optionset=TIEUnconfiguredExportDefault
```

- Synchronize item **000001**.

```
tcxml_export -item=000001 -file=itemsync.xml  
-optionset=TransferOptionSet -sync -reason=ItemIsOutDated
```

- Export the **Latest Working** revision of the **Top1** item using the **TIEConfiguredExportDefault** transfer option set.

```
tcxml_export -item=Top1 -optionset=TIEConfiguredExportDefault  
-revrule="Latest Working" -file=D:\temp\Top1_HL.xml
```

- Export a partial structure that includes only the changes to the **Top1** assembly that are tracked by configured incremental change:

```
tcxml_export -item=Top1 -targetsites=-2054508072  
-optionset=TIEConfiguredExportDefault  
-revrule="Latest Working" -sync -incrementalChangeDelta -file=/tmp/  
0001_delta.xml
```

- Fast export the objects identified in the **inp.txt** file using the **VARIANTRULE1** saved variant rule to site **56781234**.

```
tcxml_export -low_level -inputfile=d:\Temp\inp.txt  
-file=d:\Temp\top1_ll.xml  
-optionset=TIEConfiguredExportDefault -svrule=VARIANTRULE1  
-targetsites=-2054508072
```

- Fast export the **Latest Working** revisions of the objects, including suppressed BOM lines and BOM lines configured out by incremental changes.

```
tcxml_export -low_level -inputfile=d:\Temp\inp.txt  
-file=d:\Temp\top1_ll.xml -optionset=TIEConfiguredExportDefault
```

```
-revrule="Latest Working" -svrule=VARIANTRULE1 -processSuppressedOcc
-processIncrementalChanges
```

- Simulate a low-level briefcase export, validating the exported low-level TC XML and generating an error report (and no briefcase file).

```
tcxml_export -file=d:\Temp\report.html -item=root_obj_ID
-optionset=TIEConfiguredLLBCZExportDefault -briefcase
-targetsites=site_ID -dryrun
```

- Simulate a low-level briefcase export, validating the internal (pre-export) low-level TC XML and generating an error report (and no briefcase file).

```
tcxml_export -file=d:\Temp\report.html -item=root_obj_ID
-optionset=TIEConfiguredLLBCZExportDefault -briefcase
-targetsites=site_ID -dryrun=validateXMLBeforeXslt
```

- Perform a low-level briefcase export, validating the exported TC XML and creating a briefcase file if no warnings or errors are encountered. If errors or warnings are encountered, create a report file.

```
tcxml_export -file=d:\Temp\briefcase.bcz -item=root_obj_ID
-optionset=TIEConfiguredLLBCZExportDefault -briefcase
-targetsites=site_ID -validate
```

The following are site consolidation examples:

- To export the objects to a specified site.

```
tcxml_export -optionset=SiteConsolidationDefault
-targetsite=-2054508072 -inputfile=d:\input.txt
-file=d:\out.xml -low_level
```

- To synchronize the objects already exported (low level) to a specified site.

```
tcxml_export -sync -file=d:\out.xml -low_level
```

Special cases for **tcxml_export**:

- Export in-process workflow Items:
 1. Export the item in the workflow.
 2. Search for workflow job corresponding to that item.
 3. Copy this job and paste it into a new folder using a unique name.

4. Export this folder to export the job.

When a workflow process is exported with transfer ownership to the target site, the **My Worklist→Inbox→Tasks to Perform** folder at the target site does not display the task. To display the task, the affected users must delete the **Inbox** and restart their client.

Similarly, to display tasks in **ResourcePool Inbox**, delete its **Tasks to Track** and **Tasks to Perform** subfolders.

- Export a collaboration context object (CCO):
 1. In the rich client, search for the CCO.
 2. Copy the CCO and paste it into a new folder with a unique name.
 3. Export this folder to export the CCO and the entire structure context.
 4. Export the associated product structures.
- Low-level synchronization process example:
 1. Create the tables and triggers using an SQL script before exporting the data.

You can use an edited version of the following sample script, located in your **TC_ROOT/install/sitecons** directory, to manage the tables in separate tables spaces for the site consolidation tables. Before you run the script, edit the highlighted parameter values for your site.

```
sitcons_create_tablespace.sql
/* Copyright Siemens Product Lifecycle Management Software Inc.
   All Rights Reserved. */
/* This is a sample script that can be used by the administrator.
   The following parameters namely datafile, size, autoextend on maxsize,
   extent management local uniform size are the variables that need to be
   changed as required. Also, the tablespace can be named as desired */
/* Creates a separate table space to be used later for ACCT_TABLE creation */
create tablespace TCSITCONS datafile
'D:\oracle\product\10.2.0\oradata\test\TCSITCONS.dbf'
size 10M
autoextend on maxsize 100M
extent management local uniform size 64K;
/* Creates the table ACCT_TABLE using above table space */
create table ACCT_TABLE(
exp_obj_uid varchar2(15) PRIMARY KEY,
led date,
island_anchor_uid varchar2(15),
state NUMBER)
tablespace TCSITCONS;
create table SCRATCH_TABLE(
puid varchar2(15),
lsd date,
trigger_condition NUMBER)
tablespace TCSITCONS;
```

```

CREATE INDEX "SCRATCH_TABLE_INDEX" ON "SCRATCH_TABLE" (puid);
create or replace trigger fast_sync_add_trigger
before insert on PPOM_OBJECT
referencing new as newRow
for each row
BEGIN
insert into scratch_table values (:newRow.puid, :newRow.plsd, '8');
END;
/
create or replace trigger fast_sync_delete_trigger
after delete on PPOM_OBJECT
referencing old as oldRow
for each row
BEGIN
insert into scratch_table values (:oldRow.puid, :oldRow.plsd, '9');
END;
/

```

2. Export the data using the **tcxml_export** utility in low-level mode:

```

tcxml_export -optionset=SiteConsolidationDefault
-targetsite=-2054508072 -inputfile=d:\input.txt -file=d:\out.xml -low_level

```

3. Import the data at the target site using the **tcxml_import** utility in low-level mode:

```

tcxml_import -file=d:\out.xml -low_level

```

4. Copy the import-generated **out_import_results.txt** file to the source site and run the **tcxml_confirm_export** utility in low-level mode:

```

tcxml_confirm_export -file=d:\out_import_results.txt -low_level

```

5. Add, delete, and change the low-level exported data as needed.

6. Run the **tcxml_export** utility with the **-sync** argument.

```

tcxml_export -file=d:\sync.xml -low_level -sync
-optionset=SiteConsolidationDefault -inputfile=d:\input.txt

```

7. Import the generated XML file using **tcxml_import** low-level mode.

8. Repeat the earlier step to copy the import-generated **out_import_results.txt** file to the source site and run the **tcxml_confirm_export** utility in the low-level mode.

```

tcxml_confirm_export -file=d:\out_import_results.txt -low_level

```

Fast synchronization operations depend on the time stamp on the local server machine where an object is saved and edited. Therefore, modifying objects on different machines with different system times may influence the identification of out-of-sync status.

tcxml_import

Imports objects into Teamcenter from a TC XML file. The **tcxml_import** utility uses the **PS_bvr_import_new_occ** operation.

The low-level TC XML arguments of this utility provide access to site consolidation functionality. To use those arguments, you must set a license key value for the **SITCONS_AUTH_KEY** environment variable. Open a support case on Support Center to get this license key.

Caution:

Do not use the **tcxml_import** utility for importing administration data. For importing administration data, use the **admin_data_import** utility.

Property values can be updated in bulk. See *Process for updating property values in bulk*.

SYNTAX

```
tcxml_import
[-u=user-id{-p=password | -pf=password-file} -g=group]
-file=xml-file-name
[ [-xsl=xsl-file-name | -mappingcontrolfile=map-file-name]
[-errorcontinue={yes | no} ]
[ [-site=site-name]
[-transfermode=transfer-mode]
[-optionset=option-set-name] ] |
[ [-scope_rules [-scope_rules_mode=ignore | overwrite] ] ]
[-requiredLang=locale-code-1, locale-code-2, ..., locale-code-n]
[-allowedLang=locale-code-1, locale-code-2, ..., locale-code-n] ] |
[-low_level {-file=file-name}
[-bulk_load {-site=site-name -file=file-name} ]
[-tcfile_import]
[-islands=island-list]
[-bypassSiteCheck]
[-session_options=option-1:value-1, option-2:value-2,...option-n:value-n]
[-session_options_file=options-file-name]
[-briefcase]
[-dryrun [=validateXMLBeforeXslt]]
[-validate [=validateXMLBeforeXslt]]s
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the input TC XML file containing the objects to import into Teamcenter.

-xsl

Specifies the input XSL file to apply to the TC XML file before import.

-mappingcontrolfile

Specifies the mapping control file to apply to the TC XML file before import.

-errorcontinue

Indicates whether to continue the import after encountering an error. The default value is **yes**. If you specify **no** and the utility encounters an error, the utility rolls back all of the changes performed during the current import.

-site

Specifies the owning exporting site from which input TC XML data is generated. This argument is mutually exclusive with the **-scope_rules** and **-scope_rules_mode** arguments.

The argument is not valid for bulk loading a briefcase file.

-transfermode

Specifies the transfer mode name that is to be used for import. If this argument is not specified, the utility uses the **TIEImportDefault** transfer mode. This argument is mutually exclusive with the **-scope_rules** and **-scope_rules_mode** arguments.

-optionset

Specifies the option set name that contains options to use during import. This argument is mutually exclusive with the **-scope_rules** and **-scope_rules_mode** arguments.

To import a briefcase file created using bulk extract, specify the option set with the **opt_bulk_load_bcz** value set to **TRUE**.

-scope_rules

Specifies that the input XML data contains scope rules, that is, transfer modes, closure rules, filter rules, property sets, actions rules, or transfer options sets. This argument is mutually exclusive with the **-site**, **-transfermode**, and **-optionset** arguments. See *Restrictions*.

-scope_rules_mode

Specifies the import behavior when an imported rule already exists in the database. If set to **ignore**, the rule is not imported. If set to **overwrite**, the existing database rule is overwritten. If you do not specify this argument, **ignore** behavior is used. If you do not specify the **-scope_rules** argument, this argument is invalid.

-requiredLang

Specifies a list of comma separated locale values. This list is used to ensure that localized attributes in the imported data have at least one representation that can be used as the attribute primary language at the importing site. It also defines a priority order for the exporter to determine the attribute primary language. The valid locale values must match the Java locale naming convention that consists of two groups of two-character identifiers separated by an underscore character (`_`) for a particular combination of language and region. For example, **zh_CN** represents Simplified Chinese in China and **en_US** represents English in the United States.

-allowedLang

Specifies a list of comma separated locale values. This list is used to get additional representations for localized attributes in the imported data for use at the importing site. The valid locale values must match the Java locale naming convention that consists of two groups of two-character identifiers separated by an underscore character (`_`) for a particular combination of language and region. For example, **zh_CN** represents Simplified Chinese in China and **en_US** represents English in the United States.

-low_level

Performs a fast import using POM level APIs. Fast import is normally used for consolidation of site databases. You must specify only the **-file** argument; all other arguments are ignored during fast imports.

This argument provides access to site consolidation functionality and requires setting a license key value for the **SITCONS_AUTH_KEY** environment variable. Open a support case on Support Center to get this license key.

If you import 4th Generation Design (4GD) data, you must run the **appmodel_fix_scope** utility at the target site to ensure 4GD data is represented properly.

-bulk_load

Performs a fast import of legacy data from a file containing low-level TC XML formatted data or a low-level TC XML briefcase file. You must specify only the **-file** and **-site** arguments; all other arguments are ignored when bulk loading low-level data. The **-site** argument is not valid for bulk loading a briefcase file.

Be aware of the following items:

- If the **-file** argument specifies a briefcase (.bcz) file, the utility switches to low-level briefcase import mode. If you specify any optional arguments (with the exception of the **-optionset** argument) when importing a briefcase file, they are ignored. If you do not specify the **-optionset** argument, the utility uses the **BulkLoadDefault** transfer option set. If you provide a transfer option set, it must contain the **opt_bulk_load_bcz** option set to **TRUE**.
- The import uses the volume specified in the TC XML if the **TIE_Volume_Map** preference is not set or does not specify a volume mapping for the volume named in the TC XML.

If the **TIE_Volume_Map** preference specifies a volume mapping for the volume named in the TC XML, the import uses the mapped volume. If the mapped volume is unavailable, the volume specified by **opt_use_default_volume** is used. If **opt_use_default_volume** is not set to **true**, an error is returned.

-tcfile_import

Imports **ImanFile** objects using a low-level TC XML formatted GSID-based file. This argument is valid only with the **-low_level** or **-bulk_load** arguments.

-islands

Imports one or more specific islands identified by *island-list*. **-islands** is valid only with the **-low_level** and **-bulk_load** arguments. Previous islands in the TC XML file (with higher ID values) must already exist in the database for an island to be imported successfully.

island-list is a comma-delimited list of island IDs or ranges of IDs, such as "10,8,7-5,3-". If no closing of a range is provided, all of the remaining islands in the TC XML file are imported.

-bypassSiteCheck

If specified, this switch bypasses the check that prevents the utility from updating objects at the same site and the check that prevents the update of replica objects.

Note:

If you are using this utility in a Multi-Site environment, run the utility at each of the sites to update the objects belonging to the respective sites.

Users have the capability of performing updates on replica objects, eliminating the need to re-replicate all updated objects.

Use this argument only with the **-bulk_load** argument for bulk update of attributes of local objects and import of archived audit records. Because this argument requires the **-bulk_load** argument that provides access to site consolidation functionality, **SITCONS_AUTH_KEY** must be set to a license key value.

-session_options

Specifies a comma-separated list of option name-value pairs. Names and values are delimited by colons, for example:

`ContinueOnError:TRUE,GenerateReport:TRUE`

-session_options_file

Specifies an ASCII file containing session option settings using option name-value pairs. Use this option instead of the **-session_options** option when you are reusing long lists of session option name-value pairs and when values contain system-reserved special characters.

Place one name-value pair on each line of the file as follows:

```
option_1:value_1
option_2:value_2
.
.
.
option_n:value_n
```

Set an option only once in the file.

-briefcase

Specifies the imported file is formatted as a briefcase file. The file is specified with the **-file** option.

Briefcase files exported using **-bulk_extract** are not supported by **-briefcase**.

-dryrun

When importing a low level briefcase file, specifies that a simulated import be run and the imported TC XML be validated. Warnings and errors in the TC XML are reported in the file specified by **-file** (with a *.html* extension).

If **-dryrun=validateXMLBeforeXslt** is specified, the exported TC XML is validated before the XSLT is applied to it. The briefcase file is not imported in either case. **-dryrun** cannot be used with the **-validate** option.

-validate

When importing a low level briefcase file, specifies that the imported TC XML be validated. Two levels of validation occur:

- In the first validation level, configuration aspects of the briefcase file such as the user, transformation rules, data types, and the target site are validated.
- In the second pass, the objects in the briefcase file are validated.

If errors are found in the first pass, the briefcase file is not imported. If errors are found only in the second pass, the briefcase file is imported.

Whenever warnings or errors are found, they are reported in the file specified by **-file** (with a *.html* extension).

If **-validate=validateXMLBeforeXslt** is specified, the internal (pre-export) TC XML is validated instead of the exported TC XML. **-validate** cannot be used with the **-dryrun** option.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in the [Log files produced by Teamcenter](#).

RESTRICTIONS

- When importing transfer option sets, if a local option set at the exporting site is specified for import, the utility assigns the local site as the site reference and the imported option becomes local to the importing site. If a remote option at the exporting site is specified for import, the site reference of that option set is expected to exist in the database at the importing site. If it is not, the import fails. To avoid this failure, you must manually create the option set before attempting the import.
- Not all Teamcenter data is supported for import. For a list of objects that are supported, see *Teamcenter Data Exchange*.

EXAMPLES

- The following example imports objects specified by the **-file** argument:

```
tcxml_import -file=file_name.xml
```

- The following example imports objects specified by the **-file** argument according to the rules given in transfer mode specified by the **-transfermode** argument:

```
tcxml_import -file=xml-file-name -transfermode=transfer-mode-name
```

- The following example imports objects specified by the **-file** argument and uses the value specified by the **-site** argument as the exporting site for this import:

```
tcxml_import -file=file_name.xml -site=site_ID
```

- The following example imports objects specified by the **-file** argument and uses the value specified by the **-xsl** argument to apply transformation on the input XML file:

```
tcxml_import -file=file_name.xml -xsl=xsl_file
```

- The following example imports objects specified by the **-file** argument and uses the value specified by the **-errorcontinue** argument to determine whether to roll back the import if an error occurs:

```
tcxml_import -file=file_name.xml -errorcontinue=yes
```

- The following example imports objects specified by the **-file** argument and uses the value specified by the **-optionset** argument to access the options that must be used during import:

```
tcxml_import -file=file_name.xml -optionset=option_set_name
```

- The following example imports objects specified with the **-file** argument with localizable attribute values in **en_US** and **fr_FR** locales:

```
tcxml_import -file=file_name.xml  
-requiredlanguage=en_US -allowedlanguage=fr_FR
```

- The following example imports archived audit records from the file *AuditExport.xml*:

```
tcxml_import -bulk_load -bypassSiteCheck -file=c:/temp/AuditExport.xml
```

- The following example sets session option values specified in the external *sess_opts.cfg* file and imports **ImanFile** objects with physical files from the volume in the specified path:

```
tcxml_import -bulk_load -tcfile_import  
-session_options_file=c:\config\sess_opts.cfg  
-file=c:/temp0001.xml -source_volume=c:\temp\source_volume
```

- The following example explicitly sets session option values and imports objects with the **-file** argument with the specific island IDs:

```
tcxml_import -file=exportitem.xml -islands=10,8,7-5,3-  
-session_options=opt_traverse_ref_org:false,  
objForOwnXfer:1857876201:wjUx4xecAAGcRA
```

- The following example simulates a low-level briefcase import, validating the imported low-level TC XML and generating an error report (and no briefcase file).

```
tcxml_import -file=d:\Temp\report.html  
-optionset=TIEConfiguredLLBCZExportDefault -briefcase -dryrun
```

- The following example simulates a low-level briefcase import, validating the internal low-level TC XML and generating an error report without importing the briefcase file.

```
tcxml_import -file=d:\Temp\report.html  
-optionset=TIEConfiguredLLBCZExportDefault  
-briefcase -dryrun=validateXMLBeforeXslt
```

- The following example performs a low-level briefcase import, validating the exported TC XML and importing the briefcase file if no warnings or errors are encountered. If errors or warnings are encountered, a report file is created.

```
tcxml_import -file=d:\Temp\briefcase.bcz  
-optionset=TIEConfiguredLLBCZExportDefault -briefcase -validate
```

tcxml_ownership_recovery

DESCRIPTION

Changes ownership for a list of replica objects at the local site. The utility can also be used to report the current site ownership of objects related to an item. You can use this report to determine when objects imported from an unmanaged site briefcase file have incorrect owning sites.

The report contains the UIDs of the items and can be used to create the input file used to change (flip) the ownership of the file to the target site.

SYNTAX

```
tcxml_ownership_recovery -u=user-name {-p=password | -pf=password-file} [-g=group]
-action={report | flip}
{-inputfile=file-with-item-IDs | -inputuidfile=file-with-uids}
{-targetsites=site-ID | -optionset=option-set-name}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

When Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO, rather than being authenticated against the database.

- If you do not supply these arguments, the utility attempts to join an existing SSO session.
- If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. The file must be a single-line ASCII file containing the password in clear text. Teamcenter Environment Manager prompts you for a password and creates the password file during installation.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

Specifies whether to create a report file showing the site ownership of the input file items and their related objects or to perform site ownership change on the input file items.

report

Generates a file that displays the owning site for the input file objects and the objects related to them.

The report file is in the same directory as the input objects file defined by the **-inputfile** argument. The file name follows the format: *input-file-name_report.log*.

When you specify a report action:

- The **-inputfile** argument is required.
- The **-optionset** argument is optional.
- The **-inputuidfile** and **-targetsites** arguments are ignored.

flip

Transfers the ownership of the objects in the input file to the target site. The input file is defined by the **-inputuid** argument.

When you specify a flip action:

- The **-inputuidfile** and **-targetsites** arguments are required.
- The **-inputfile** and **-optionset** arguments are ignored.

-inputfile

Specifies the file containing a list of item IDs for which you want to determine the owning site. The file must contain item IDs on separate lines.

This argument is ignored for flip actions.

-inputuidfile

Specifies the file containing a list of object UIDs for which you want to change the owning site to the target site. The target site is identified by the **-targetsites** argument. The file must contain object UIDs on separate lines.

This argument is ignored for report actions.

-targetsite

Specifies the new owning site of the objects defined in the input file. The input file is identified by the **-inputuidfile** argument.

This argument is ignored for report actions.

-optionset

Specifies the name of the transfer option set (TOS) used to determine the objects related to the item IDs that are included in the report file.

This argument is ignored for flip actions.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter* Log files produced by Teamcenter.

RESTRICTIONS

None.

EXAMPLES

- To generate a report file containing the items and their related objects listed in the *check_itemids.txt* file:

```
tcxml_ownership_recovery -u=Tc-admin-user -p=password -g=group
                        -action=report
                        -inputfile=check_itemids.txt
```

- To generate a report file containing the items and their related objects as determined by the default site consolidation TOS:

```
tcxml_ownership_recovery -u=Tc-admin-user -p=password -g=group
                        -action=report -inputfile=itemids.txt
                        -optionset=SiteConsolidationDefault
```

- To transfer the ownership of the objects listed by UIDs in the *object_uids.txt* file to site **-100004379**:

```
tcxml_ownership_recovery -u=Tc-admin-user -p=password -g=group  
-action=flip  
-inputuidfile=object_uids.txt -targetsites=-100004379
```

tcxml_validate

Validates the contents of a low-level TC XML formatted file. This file may be an a file exported from a Teamcenter site or constructed by a custom solution that maps legacy system data to the Teamcenter unified data model. You can use this to validate the file information prior to importing it into a Teamcenter site or to compare the Teamcenter data created by importing the file to the contents of the imported file. This utility requires a license key. For information about obtaining and setting the license key, see [Support Center](#).

SYNTAX

```
tcxml_validate [-u=user-id {-p=password | -pf=password-file} -g=group]
{-file=input-xml-file -action={pre | post}}
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the input TC XML file name. The value can be either an absolute path (full path name) or a relative path name.

-action

When set to **pre**, checks the named file for conformance to the importing sites schema, logical data model, and business constraints. The output of the utility is written to a log file that uses the naming convention *input-xml-file-name_validator.log*. This file uses a pipe (|) character as a field delimiter and can be imported into Microsoft Excel for easy reading. This file contains any errors and discrepancies found by the utility.

Note:

The **post** action is not available at this time. Use the **pre** action.

When set to **post** compares the named file contents to the data created when the file was imported to verify the objects where imported properly.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

You must acquire a license key from Siemens Support and set the **SITCONS_AUTH_KEY** environment variable to this value to use this utility.

EXAMPLES

The following command validates a file (**0001.xml**) and creates a log file (**0001_validator.log**) containing the results of the validation.

```
tcxml_validate -u=Tc-admin-user -p=password -g=group -file=D:\temp\0001.xml
               -action=pre
```

tcxml_xfer_ownership

DESCRIPTION

Transfers the ownership of a set of Teamcenter objects. You can use this utility to change ownership of all objects owned by a site during site consolidation, change ownership of related objects during transition from Teamcenter Enterprise to Teamcenter (flip-the-switch ownership transfer), or change object ownership when importing objects into Teamcenter from another data management system.

For performance reasons during site consolidation, this utility changes ownership of objects using SQL calls, not the POM API. Rich client refresh works only when POM calls are used to change an attribute, including ownership changes. Therefore, ownership changes that the utility enacts during an active rich client session are not reflected in the interface. You must restart all active rich clients to get the changes. This is normally not an issue because user access must be prohibited during the *time critical period* when this utility is used.

A valid license key is required to use this utility for site consolidation ownership changes.

The **perform_highlevel** argument enables the flip-the-switch ownership transfer feature of this utility. By default, during flip-the-switch ownership transfer, this utility uses the POM API to change object ownership. Therefore, ownership changes that the utility enacts during an active rich client session are reflected in the interface. You are not required to restart the active rich clients to get the changes.

For better performance, you can set the **TIE_flip_using_SQL** preference to **true** to use SQL calls during flip-the-switch ownership transfers. If you use this preference setting, you must restart all active rich clients to get the changes.

A valid license key is not required to use this utility when you enable the flip-the-switch feature.

SYNTAX

```
tcxml_xfer_ownership -u=user-name {-p=password | -pf=password-file} [-g=group]
  -action={extract | perform | update_status |
perform_highlevel | perform_highlevel_dryrun}
[-inputfile=file-with-object-list]
[-change_ownership_to=site-ID]
[-dryrun [-startDate=MM/DD/YYYY -endDate=MM/DD/YYYY] [-source_extinct] ]
[-file=status-output-file]
[-report=report-file-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

When Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO, rather than being authenticated against the database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. The file must be a single-line ASCII file containing the password in clear text. Teamcenter Environment Manager prompts you for a password and creates the password file during installation.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

Specifies whether to extract, perform, or update status.

To use the **extract**, **perform**, or **update_status** values, the **SITCONS_AUTH_KEY** environment variable must be set to a valid key value. Open a support case on Support Center to get this license key.

extract

Generates the file for ownership transfer.

The **-change_ownership_to**, **-report**, or **-dryrun** arguments are supported for this action.

perform

Transfers the ownership at target site.

The **-inputfile**, **-file**, **-source_extinct**, or **-dryrun** arguments are supported for this action.

update_status

Updates the status at source site.

The **-inputfile**, **-change_ownership_to**, **-source_extinct**, **-report**, or **-dryrun** arguments are supported for this action.

perform_highlevel

Performs a high-level (flip-the-switch) ownership transfer at the target site

The **-inputfile** and **-change_ownership_to** arguments are supported for this action. Any other arguments are ignored.

This action generates two outputs files, the *input-file-name*_flip_objects.log file lists all objects transferred and the *input-file-name*_results.xml file contains the flip status of objects given in input file.

perform_highlevel_dryrun

Generates a report to preview the ownership change of objects transferred by a high-level (flip-the-switch) ownership transfer at the target site.

The **-inputfile** and **-change_ownership_to** arguments are supported for this action. Any other arguments are ignored.

-inputfile

Specifies the name of the file containing the objects for ownership change.

-change_ownership_to

Specifies the site ID of the target site. This argument is required when the utility is run at the source site. If this argument is omitted, the utility uses the local site as the target site. If either the **-dryrun** or the **perform_highlevel_dryrun** argument is specified, the ownership transfer is not performed.

-dryrun

Generates a report to preview the ownership change of objects transferred by the site consolidation tools, and identifies inconsistencies.

To preview the objects at the target site that are owned by the source site and are inconsistent with the source site due to modifications made to them at the target site post-replication, set the **TIE_DRYRUN_VALIDATION=TRUE** preference at the target site.

The **-startDate** and **-endDate** arguments are valid only with the **-dryrun** argument. The utility generates a dry run report for objects transferred within the date range specified by the **-startDate** and **-endDate** values.

To use the dry run mode to check objects of an island owned by the source site that are not covered in the accountability table for ownership transfer, set the **TIE_DRYRUN_VALIDATION** preference to **TRUE** at the target site.

For debug information, set the following environment variables:

TC_SLOW_SQL=1

TC_SQL_DEBUG=BJPT

-startDate

Specifies the starting date for replicas and inconsistencies reported by the **-dryrun** argument. This argument is used in conjunction with the **-endDate** argument and is ignored if the **-dryrun** argument is not specified. The date supplied must be in supplied in a *MM/DD/YYYY* format.

-endDate

Specifies the ending date for replicas and inconsistencies reported by the **-dryrun** argument. This argument is used in conjunction with the **-startDate** argument and is ignored if the **-dryrun** argument is not specified. The date supplied must be in supplied in a *MM/DD/YYYY* format.

-source_extinct

Generates the list of replica items at the target site which are owned by the source site and which were not transferred by site consolidation tools.

This argument can be used only when the **-dryrun** argument is used and is ignored otherwise.

-file

Specifies the name of the output file (with an absolute path) that contains the status of ownership transfer of the objects at the target site. This argument is valid only for the site consolidation **-action=perform** argument.

-report

Specifies the name of the report file or extract file. This argument is valid only for site consolidation actions.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To run at the source site and extract to a file the objects for ownership transfer:

```
tcxml_xfer_ownership -u=Tc-admin-user -p=password -g=group
-action=extract -report=D:\Temp\ownership_transfer_extract.txt
-change_ownership_to=target_site_id [-dryrun]
```

- To transfer the ownership or validate the inconsistencies for ownership transfer at the target site:

```
tcxml_xfer_ownership -u=Tc-admin-user -p=password -g=group
-action=perfrom -inputfile=ownership_transfer_extract.txt
-file=ownership_transfer_status.txt [-dryrun] [-source_extinct]
```

- To update the ownership transfer status or to generate report for a dry run at the source site:

```
tcxml_xfer_ownership -u=Tc-admin-user -p=password -g=group
-action=update_status -inputfile=ownership_transfer_status.txt
-report=ownership_transfer_update_status.txt
-change_ownership_to=target_site_id [-dryrun]
```

- To use flip-the-switch ownership transfer to change ownership at the target site:

```
tcxml_xfer_ownership -u=Tc-admin-user -p=password -g=group-
action=perform_highlevel -inputfile=flip_the_swich_input.txt
-change_ownership_to=target_site_id
```

- To use flip-the-switch ownership transfer to determine ownership change at the target site without actually changing ownership:

```
tcxml_xfer_ownership -u=Tc-admin-user -p=password -g=group
-action=perform_highlevel_dryrun -inputfile=flip_the_swich_input.txt
-change_ownership_to=target_site_id
```

- The following example shows the contents of a sample file used as the input for an ownership transfer (-inputfile=flip_the_switch_input.txt):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- flip_the_switch_input.txt file -->
<TcFlipXML
  author="super user" date="2010-03-24" time="16:10:50" language="en"
  schemaVersion="2.0" >
  <GSIdentity class="Cmponent" context="" factor="" atomic="false"
    label="udzjduesvsun0sugms2--aRZ" split_token="" sub-label="" system="213456789"
    transient_island_id="udzjdvcsvsun0sugms2--aRZ" elemId="1" />
  <GSIdentity class="CmpnMstr" context="" factor="" atomic="false"
    label="udzjdvcsvsun0sugms2--aRZ" split_token="" sub-label="" system="213456789"
    transient_island_id="udzjdvcsvsun0sugms2--aRZ" elemId="2" />
  <GSIdentity class="Assembly" context="" factor="" atomic="false"
    label="udzjdwksvsun0sugms2--aRZ" split_token="" sub-label="" system="213456789"
    transient_island_id="udzjdwksvsun0sugms2--aRZ" elemId="3" />
  <GSIdentity class="AssmMstr" context="" factor="" atomic="false"
    label="udzjdxisvsun0sugms2--aRZ" split_token="" sub-label="" system="213456789"
    transient_island_id="udzjdwksvsun0sugms2--aRZ" elemId="4" />
</TcFlipXML>
```

You can construct the input file manually by copying the **GSIdentity** elements from the TC XML files used for importing the objects.

transaction_monitor

Deletes a designated range of the records being monitored using Data Exchange Transactions in Active Workspace.

SYNTAX

```
transaction_monitor -u=user-ID {-p=password | -pf=password-file} -g=group
-function={delete}
-from="start_date" -to="end_date"
[-final_status={Success | Fail}]
[-filename="input_file_name"]
[-dryrun]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If used without a value, the user's default group is assumed.

-f=delete

Deletes the records in the range specified with the **-from** and **-to** arguments.

-from

Specifies the starting date (inclusively) for deleting transaction records. Use this argument with the **-to** argument. The format of the date is "YYYY-MM-DD HH:MM:SS", with *HH* being an hour from **00** to **24**. Place the value inside double quotes because of the space between the day and the hour.

Not valid when using the **-filename** argument.

-to

Specifies the ending date (inclusively) for deleting transaction records. Use this argument with the **-from** argument. The format of the date is "YYYY-MM-DD HH:MM:SS", with *HH* being an hour from **00** to **24**. Place the value inside double quotes because of the space between the day and the hour.

Not valid when using the **-filename** argument.

-final_status

Deletes only those transaction records with the specified status value. Not valid when using the **-filename** argument. If not provided, all transactions within the date range are deleted.

-filename

Specifies the path and file containing a list of transaction record ID values to be deleted. Each line in the file must contain one ID value. Enclose the value in double quote marks when the path or file name contains spaces.

-dryrun

Runs the command without deleting any transaction records. Displays the transaction records deleted if the command is run without the **-dryrun** option.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment..

FILES

As specified in Log files produced by Teamcenter.

EXAMPLES

- Delete transaction records in a date range:

```
transaction_monitor -u=userID -p=password -g=group
-f=delete -from="2020-09-01 15:00:00" -to="2020-09-30 15:00:00"
```

- Delete transaction records in a date range with a final status of **Success**:

```
transaction_monitor -u=userID -p=password -g=group
-f=delete -final_status=Success -from="2020-09-01 15:00:00" -
to="2020-09-30 15:00:00"
```

- Delete transaction records with a file containing TxnRecord object UIDs:

```
transaction_monitor -u=userID -p=password -g=group
-f=delete -f=delete -filename="input_file_name"
```

upload_plmxml_struct

Uses the item ID and revision ID of the top node of the assembly and the revision rule and executes the **bomwriter** utility with options specified in the arguments. The PLM XML file generated is then attached as a named reference to the **DirectModelAssembly** dataset.

SYNTAX

```
upload_plmxml_struct [-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
-rev=revision-id -rev_rule=revision-rule [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the item ID of the top node of the assembly structure.

-key

Specifies the key of the top node of the assembly structure.

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-rev

Specifies the revision ID of the top node of the assembly structure.

-rev_rule

Specifies the revision rule to use to apply on the assembly and create the structure output.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Open the Teamcenter menu shell with the database connection variables set and execute the following command:

```
upload_plmxml_struct -u=Tc-admin-user -p=password -g=group -item=000125
  -rev=001 -rev_rule="Latest Working"
```

utility_execution_set

This utility runs supported Teamcenter utility commands under a single login session.

utility_execution_set logs in once, calls all utility commands contained in a referenced text file in the order specified, and then logs out of the session.

None of the called utility commands log in and log out; they all run in the login session created by **utility_execution_set**. Hence, execution time for the set is reduced because there is only one log in and one log out for the set.

All the commands in the list must use the same login credentials (user name, password, and group); successive commands run in the session that is created for the first command in the list.

Supported utilities include the following:

```
aie_install_datamodel_objs
import_file
install_callback
install_handlers
install_xml_stylesheet_datasets
l10n_cots_import_translations
plmxml_import
plmxml_tm_edit_xsl
preferences_manager (for -set_overlay use -loginType=POM)
tcxml_import
```

SYNTAX

```
utility_execution_set
  -loginType=<POM, ITK, AUTO>
  -configFile=<Path and name of file containing set of utilities to execute>
  [-forceUnixEnvFormat]
  [-log=<Path and name of output log file>]
  [-h]
```

ARGUMENTS

-loginType

All utility command lines specified in with **-configFile** use this type of login for the single session. Use **AUTO** as the input for this utility except as noted in the list of supported utilities.

Valid values:

POM Indicates using a POM session (which is restricted to POM calls)

ITK Indicates using a regular ITK session

AUTO Indicates using an autologin session

-configFile

Each line in this file contains the command line used for utility execution. The format is the same as used for inclusion in the *install_project.default* and *upgrade_project.default* files maintained in the Business Modeler IDE.

Environment variable specification is allowed in the command line(s) specified in this file.

Windows platform uses a specification like %<env var name>%.

Example:

```
%TC_PASSWD%
```

Linux platforms use a specification like \${<env var name>}.

Example:

```
${TC_PASSWD}
```

-forceUnixEnvFormat

Force this utility to use the UNIX format specification to determine environment variable values in the file specified by **-configFile**.

-log

The path of the log file which contains results of this execution.

-h

Displays the help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment.

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

This utility is limited to use with the supported utilities.

EXAMPLE

Run the commands in the file *my_preferences_commands.txt*.

```
utility_execution_set -loginType=AUTO -configFile=my_references_commands.txt
```

Contents of *my_preferences_commands.txt*:

```
preferences_manager -u=Tc-admin-user -p=password -g=group -mode=import -action=OVERRIDE
  -scope=SITE -context=Teamcenter
  -file=%TC_INSTALL_DIR%/cntmgt10d40/cntmgt10d40_preference_override.xml
tcxml_import -u=Tc-admin-user -p=password -g=group -file=%TC_INSTALL_DIR%/
systemsengineering/systemsengineering_transfermodes.xml -scope_rules
  -scope_rules_mode=overwrite
tcxml_import -u=Tc-admin-user -p=password -g=group -scope_rules -scope_rules_mode=append
  -file=%TC_INSTALL_DIR%/systemsengineering/systemsengineeringTransferOptionSets.xml
preferences_manager -u=Tc-admin-user -p=password -g=group -mode=import -action=OVERRIDE
  -scope=SITE -context=Teamcenter
  -file=%TC_INSTALL_DIR%/systemsengineering/systemsengineering_preference_override.xml
```

validate_and_replicate_assembly

Validates an assembly with checks based on parameters and generates a PLM XML file containing a list of all components in a configured assembly. The utility gets a list of participating sites using the project that the assembly root item is assigned to and invokes the replication mechanism to update the assembly components to all the participating sites. Use this utility only from a translator task to generate PLM XML output for an assembly that is being released.

SYNTAX

```
validate_and_replicate_assembly [-u=user-name {-p=password | -pf=password-file}
-g=group] -item_id=assembly-root -rev=revision -revision_rule=revision-rule
[-variant_rule=rule-to-configure-BOM] [-check_precise] [-check_no_stubs]
[-check_all_released] [-h]
```

ARGUMENTS

Note:

Entries in parentheses are accepted abbreviations for arguments.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

If this argument is not used, the system assumes the *user-ID* value is the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

Caution:

For HTTP enabled sites, remote site operations log on using the default group for the user supplied with the **-u** argument. Any value supplied with the **-g** argument is ignored.

-item_id

Specifies the top item in the assembly structure.

-rev

Specifies the top item revision used to configure the structure.

-revision_rule

Specifies the name of the revision rule used to configure the assembly.

-variant_rule

Specifies the variant rule used to configure the assembly. If not specified, the default variant rule set at the site level is used.

-check_precise

Performs a validation check on the complete assembly to ensure that all sub assemblies are precise.

-check_no_stubs

Performs a validation check on the complete assembly to ensure that there are no stubs in the assembly.

-check_all_released

Performs a validation check on the entire assembly to ensure that all components are released.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

EXAMPLES

Note:

Required logon information is omitted from the following examples.

The following command validates that the **assy_root** assembly is a precise assembly with no stubs and all components are released. It also replicates the assembly to all participating sites:

```
validate_and_replicate_assembly -u= -p= -g=dba -item_id=assy_root -rev=A  
-revision_rule=Latest Released -check_precise -check_no_stubs  
-check_all_released
```


8. Visualization utilities

harvest_mmv_index

Creates or updates the spatial cell index for a particular product specified by an item revision. The spatial cell index describes the spatial structure of the massive product data and is used by viewer clients to quickly load and render the product geometry.

The utility traverses the product structure and locates all unconfigured occurrences that contain a JT dataset. Each occurrence is identified by the revision independent path from the root to this occurrence. The bounding box information is computed for each occurrence, and the bounding box is rolled up to the root coordinate system by applying the accumulated transform. The rolled-up bounding boxes are used to create a spatial tree, which is saved as an **.mmv** file. This **.mmv** file is uploaded as a named reference of a **Fnd0SpatialHierarchy** dataset.

Syntax

```
harvest_mmv_index [-u=user-id {-p=password | -pf=password-file} -g=group]
[-item=item-id | -key=item-key | -itemid=root-item][--rev=revision] [--rev_rule=revisionRuleList][--log=file-name][--purge][--h]
```

Arguments

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. The file must be a single-line ASCII file containing the password in clear text. Teamcenter Environment Manager prompts you for a password and creates the password file during installation.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-item

Specifies the item ID for the root of the BOM structure.

-key

Specifies a string used to identify an item instead of specifying an item using the **-item** and **-rev** arguments.

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-itemuid

Specifies the UID to identify the item of the product to be harvested. If this option is specified, the **-item** and **-key** options cannot be specified.

-rev

Specifies the item revision for the root of the BOM structure.

-rev_rule

Specifies the revision rules that are supplied for the product that is to be harvested. Each specified revision rule is delimited by a comma. If no revision rule is supplied, the product is harvested as unconfigured.

-log

Specifies the name for the log file. By default, the log file is *harvest_spatial_index_time.log*.

-purge

Purges the MMV indexes for all products. Purging indexes periodically keeps database tables from becoming too large, which can degrade harvesting and system performance.

Note:

Once this option is used, the MMV indexes for products must be rebuilt by running the utility again.

-h

Displays help for this utility.

Environment

As specified in *Setting up a Teamcenter command-line environment*.

Files

As specified in *Log files produced by Teamcenter*.

Restrictions

None.

Examples

- To harvest a specific revision of a product (**item_id=000004**), configured with revision rules **Latest Working** and **Latest Released**, type the following command on a single line:

```
harvest_mmv_index -u=adminjones -p=passjones -g=admin -item=000004  
-rev=001  
-rev_rule="Latest Working, Latest Released"
```

- To harvest spatial cell indexes for a specific revision of an unconfigured product (**item_id=000004**) using the multifield key, type the following command on a single line:

```
harvest_mmv_index -u=adminjones -p=passjones -g=admin -key=000004-key  
-rev=001
```




9. Manufacturing utilities

acc_check_client

Logs onto Teamcenter, performs an accountability check, stores the results in occurrence groups and/or an Excel report and logs off. You can run this utility from a Teamcenter command window or from a batch file that is developed to perform a specific check on two specific structures. When running Teamcenter in a two-tier environment, the batch file normally starts its own server.

For more information about running accountability checks in batch mode, see *Manufacturing Process Planning — Deployment and Administration*.

SYNTAX

```
acc_check_client -u=user-id {-p=encrypted-password | -pf=password-file} [-g=group]
-host="server-connection-url-or-IIOP"
-sourceSC=source-structure-context-name|-sourceBVR=source-BOM-view-revision
-targetSC=target-structure-context-name|-targetBVR=target-BOM-view-revision
-sourceClosureRule=source-closure-rule
-targetClosureRule=target-closure-rule
[-occGroupName=name-of-occurrence-group]
[-createOccGroupOnTarget=occurrence-group-output-to-target]
[-reportName=dataset-name-containing-Excel-report]
-check="full,missingTarget,missingSource,multiple"
-partialMatchProperties="list-of-properties"
[-sourceRevisionRule=source-revision-rule]
[-targetRevisionRule=target-revision-rule]
[-sourceVariantRule=source-variant-rule]
[-targetVariantRule=target-variant-rule]
[-sourceDateEff=source-effectivity-date]
[-targetDateEff=target-effectivity-date]
[-showUnconfigured="variants,changes,assignments"]
[-skipScopeLines=source-and-target-scope-lines-to-be-excluded-from-results]
[-showResults]
[-descrip=server-login-discriminator-value]
[-loginRetries=number-of-times-client-will-attempt-to-login]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

The user needs appropriate write access to the source and target structures in order to create occurrences groups and attach reports.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password. This password must be encrypted

You can obtain an encrypted password by typing:

```
ipa_b_executer -p=plain-text-password -encrypt
```

Teamcenter returns the encrypted password as output.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-host

Specifies server connection information. A URL or an IIOP must be specified.

Four-tier example:

```
"http://server:port/tc"
```

Two-tier example:

```
"corbaloc:iiop:localhost:7777/localserver7777"
```

Quotation marks must be included.

-sourceSC

Specifies the unique name of a structure context in the database that contains the source structure. Teamcenter uses this value to open the source structure. Teamcenter requires either a source

structure context specified in this argument or a source BOM view revision specified in the **-sourceBVR** argument.

-sourceBVR

Specifies the UID (internal unique identifier) of the BOM view revision for opening the source structure. Teamcenter requires either a source structure context specified in the **-sourceSC** argument or a source BOM view revision specified in this argument.

-targetSC

Specifies the unique name of the structure context in the database that contains the target structure. Teamcenter uses this value to open the target structure. Teamcenter requires either a target structure context specified in this argument or a target BOM view revision specified in the **-targetBVR** argument.

-targetBVR

Specifies the UID (internal unique identifier) of the BOM view revision for opening the target structure. Teamcenter requires either a target structure context specified in the **-targetSC** argument or a target BOM view revision specified in this argument.

-sourceClosureRule

Specifies the filtering rule for the source structure.

-targetClosureRule

Specifies the filtering rule for the target structure.

-occGroupName

Specifies the name of the occurrence group that is created or updated on the source structure. If no name is specified, no groups are created or modified.

-createOccGroupOnTarget

Specifies the occurrence group output to the target. If the **missingSource** check is specified, the output to target is automatically turned on.

-reportName

Specifies the name of the dataset to which the Excel report is attached. If no name is specified, no report is generated.

-check

Specifies the type of result the check reports. This argument can have one or more of the following comma-separated values:

full,missingTarget,missingSource,multiple

-partialMatchProperties

Specifies the internal property names to be checked for partial match. This argument can contain multiple property names separated by a comma.

-sourceRevisionRule

Specifies the revision rule on the source structure. The default is determined by the source structure context or by a system setting.

-targetRevisionRule

Specifies the revision rule on the target structure. The default is determined by the target structure context or by a system setting.

-sourceVariantRule

Specifies the classic variant rule on the source structure. The default is determined by the source structure context or no rule.

-targetVariantRule

Specifies the classic variant rule on the target structure. The default is determined by the target structure context or no rule.

-sourceDateEff

Specifies the GMT date for the effectivity on the source structure. If no date is specified, the effectivity from the source structure context is used. The format is *yyyy-MM-ddThh:mm:ss zz:zz*. For example:

2005-05-20T14:32:05-08:00

-08:00 represents minus 8 hours from GMT (resulting in Pacific Standard Time)

-targetDateEff

Specifies the GMT date for the effectivity on the target structure.

If no date is specified, the effectivity from the target structure context is used.

-showUnconfigured

Lists which unconfigured occurrences to show or hide on both structures. If this argument is not specified, all unconfigured lines are hidden.

Valid values are one or more of the following:

variants,changes,assignments

-skipScopeLines

Specifies that the source and target scope lines should not be included in the results.

-showResults

Prints accountability check results statistics. If this argument is not included, result statistics are not printed.

-descrip

Specifies a value to be used as the server login discriminator. If not provided, the client will generate a random string to be used as the server login discriminator.

-loginRetries

Specifies the maximum number of times the client application will attempt to login.

The default value is 30 at 6-second intervals. The value must be greater than 0.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLE

This example shows the input for an accountability check that searches for lines that are missing in the target structure. It outputs a Microsoft Excel report called **TestReport** that displays the results.

Input:

```
acc_check_client -u="tcadm" -p="tcadm-password"
-sourceClosureRule="AccountabilityAll" -targetClosureRule="AccountabilityAll" -
host="corbaloc:iiop:localhost:7777/localserver7777" -check="missingTarget"
-reportName="TestReport" -sourceBVR="QTZJ29zQAABaaA" -targetBVR="g7UJ29zQAABaaA"
-u=tcadm
-p=tcadm-password
-sourceClosureRule=AccountabilityAll
-targetClosureRule=AccountabilityAll
-host=corbaloc:iiop:localhost:7777/localserver7777
-check=missingTarget
-reportName=TestReport
-sourceBVR=QTZJ29zQAABaaA
-targetBVR=g7UJ29zQAABaaA
```

Output:

Accountability Check Report		
Date	29/10/2013	
Time	09:45:32	
User	infodba	
Source filtering rule	AccountabilityAll	
Limit depth in source	-1	
Target filtering rule	AccountabilityAll	
Limit depth in target	-1	
Additional equivalence criteria		
Compare additional criteria		
Incremental change		
Source top level item	013038/A;1-topltem (View)	
Source revision rule	Latest Working	
Source variant rule	Unspecified	
Source context		
Source Line	013038/A;1-topltem (View)	
Target top level item	013048/A;1-mbom (View)	
Target revision rule	Latest Working	
Target variant rule	Unspecified	
Target context		
Target Line	013048/A;1-mbom (View)	
Checked Objects		
Source	Target	Match
013038/A;1-topltem (View)	***Not Found***	X
013040/A;1-e-assy (View)		
Accessory Pulley/A;1	***Not Found***	X
...		
...		

accountability_check

Runs the equivalent of an advanced accountability check and generates Microsoft Excel and occurrence group reports in a two-tier environment.

For more information, see *Manufacturing Process Planning — Deployment and Administration*.

SYNTAX

```
accountability_check -u=user-id -p=password | -pf=password-file] [-g=group ][acSetting=setting-  
dataset-uid][-sourceSC=source-structure-context-uid] | [-sourceRootId=source-root-item-id] [-  
sourceRevId=source-revision-id] [-targetSC=target-structure-context-uid] | [-targetRootId=target-root-  
item-id] [-targetRevId=target-revision-id]  
[-sourceRevRule=source-revision-rule][-targetRevRule=target-revision-rule]  
[-sourceScope=source-scope-item id][targetScope=target-scope-item-id]  
-acSetting=accountability-check-setting  
[-sourceSC=source-structure-context|-sourceRootId=source-root-id]  
[-sourceRevId=source-revision-id]  
[-targetSC=target-structure-context|targetRootId=target-root-id]  
[-targetRevId=target-revision-id]  
-ogName=occurrence-group-name|  
-createOGOnTarget=create-occurrence-group-on-target  
-reportName=excel-report-name  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-acSetting

Specifies either the uid of a saved setting dataset or the absolute path to a setting XML file on the local disk. This argument is required.

-sourceSC

Specifies the UID of a **StructureContext** object for the source structure. Either **sourceSC** or **sourceRootId** is required.

-targetSC

Specifies the UID of a **StructureContext** object for the target structure. Either **targetSC** or **targetRootId** is required.

-sourceRootId

Specifies the root item's item ID or multi-site key for the source structure. Either a **sourceSC** or **sourceRootId** is required. If **sourceRootId** is used, **sourceRevRule** is required.

-targetRootId

Specifies the root item's item ID or key for the target structure. Either a **targetSC** or **targetRootId** is required. If **targetRootId** is used, **targetRevRule** is required.

-sourceRevRule

Specifies the BOM window's revision rule name to configure the source structure. Required if **sourceRootId** is specified.

-targetRevRule

Specifies the BOM window's revision rule name to configure the target structure. Required if **targetRootId** is specified.

-sourceRevId

Specifies the root item's revision to configure the source structure. Optionally used with **sourceRootId**.

-targetRevId

Specifies the root item's revision to configure the target structure. Optionally used with **targetRootId**.

-sourceScope

Specifies the item ID, key, or ID in context (IDIC) of the source scope line, where child lines are traversed. If specified, **sourceScope** is used as the scope of the source structure for the accountability check. Otherwise, the root line is used as the scope of the source structure.

-targetScope

Specifies the item ID, key, or ID in context (IDIC) of the target scope line, where child lines are traversed. If specified, **targetScope** is used as the scope of the target structure for the accountability check. Otherwise, the root line is used as the scope of the target structure.

-ogName

(Optional) Specifies the name of the occurrence group to be created or updated. If specified, **ogName** overwrites the value of the **resultName** element in the accountability check setting.

Note:

Creating an occurrence group on a generic or product BOP structure is not supported.

-createOGOnTarget

(Optional) Specifies that an occurrence group be created on the target structure.

Note:

Creating an occurrence group on a generic or product BOP structure is not supported.

-reportName

Specifies the name of the dataset to which the Excel report is attached. If specified, **reportName** overwrites the value of the **reportName** element in the accountability check setting.

-h

Displays help for this utility.

assy_jt_creator

Creates a JT representation of a given Teamcenter assembly. The resulting file is stored in the file system and, optionally, in the database.

You can use the utility to collect the JT files on each node in a configured structure and combine all JT files into a single monolithic assembly JT. You can use a monolithic JT file to view assemblies in Resource Browser.

The utility can also create an assembly JT file that contains pointers to other JT files, each representing a component in an assembly. Each node can have a JT file and a transformation of that node.

Additionally, the utility can create a JT file for use with Tecnomatix eMServer only.

SYNTAX

```
assy_jt_creator -u=user-id -p={password | -pf=password-file} [-g=group ]
{-sc=structure-context |
{ -item=item-id | -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN] }}
-tar_loc=target-location
[-i=batch-file]
[-log=logfile-name]
[-rev=revision]
-rev_rule=revision-rule
[-var_rule=variant-rule]
[-tmode=transfer-mode-name]
[-jt_type={1 | 2 | 3} ]
[-pin=0 | 1]
[-cof=0 | 1]
[-dt_type=dataset-type] [-dt_rel=dataset-relation]
[-dt_ref=named-reference]
[-de=n | e | a | r]
[-assy_update={create | replace} ]
[-skip_root_dset_with_name=dataset-name]
[-skip_remote_lines]
[-debug]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

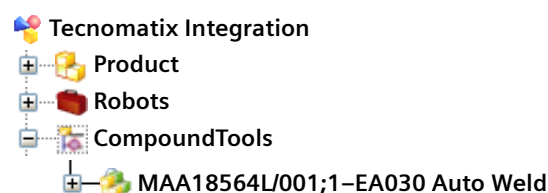
-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-sc

Specifies the name of the structure context (see **CompoundTools** in the following figure). If you include this argument, item ID, revision ID, revision rule, variant rule, and transfer mode arguments are not required.



For more information about structure contexts, see [Multi-Structure Management](#).

-item

Specifies the item ID of the top line.

If you do not include the **-sc** argument, either this argument or the **-key** argument is required.

-key

Specifies the key of the top line when multiple attributes are used to form the unique item ID. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-tar_loc

Specifies the full operating system path to an existing location where Teamcenter stores the JT assembly file.

This argument is required.

-i

Specifies an input batch file. This XML file uses elements that are equivalent to the command line arguments and contains information about the assembly JT files you want included in the monolithic JT assembly. Use this method to simplify the input of parameters. This allows you save a set of arguments you can use each time you run the utility.

This file must contain the following elements as a minimum (all defaults are used in this file):

Note:

For more information, see the equivalent command line argument description (identified in comments in the file listings).

```
<ParamBatch> <!-- Required root element -->
  <ParamSet id=""> <!-- Element holding JT file parameters -->
    <!--If a <StuctureContext> element is not included, the
  <ItemId>
    or <KeyId> element is required. See next file listing. -->
    <StructureContext> </StructureContext> <!-- See -sc argument -->
    <TargetLoc> </TargetLoc> <!-- See -tar_loc argument -->
    <DatasetType> </DatasetType> <!-- See -dt_type argument -->
  </ParamSet>
</ParamBatch>
```

The **ParamBatch** element contains all **ParamSet** elements. A separate **ParamSet** element with a unique **id** attribute is required for each JT file you include in the assembly.

This file shows all possible elements that you can specify with empty elements for informational purposes. No defaults are used. Unless an element contains only attributes, which is not the case for an input batch file, well-formed XML requires that you provide a value between the element opening and closing tag. Therefore, do not include empty elements if you want to use the default value.

```
<ParamBatch>
  <ParamSet id="">
    <ItemId> </ItemId> <!-- See -item argument -->
    <!-- If multiple key attributes (multifield key) are used, the <KeyId>
```

```

element replaces the <ItemId> element. See -key
-->
<RevId> </RevId> <!-- See -rev argument -->

<RevisionRule> </RevisionRule> <!-- See -rev_rule argument -->
<!-- If the revision rule is not specified, the utility uses the default
      "Latest Working" revision rule.
-->

<VariantRule> </VariantRule> <!-- See -var_rule argument -->

<TransferMode> </TransferMode> <!-- See -tmode argument -->
<!-- If TransferMode is not specified, the utility uses the default
      "JTDataExportDefault" transfer mode -->

<JtAssyType> </JtAssyType> <!-- See -jt_type argument -->
<!-- The type of assembly to be created can be specified as:
      1 - Monolithic JT
      2 - Assembly JT
      3 - CoJT.
      The utility uses "Monolithic" if this is not specified
-->

<ProcessIntermediateNode> </ProcessIntermediateNode>
<!-- See -pin argument -->
<!-- ProcessIntermediateNode indicates if the intermediate nodes need to be
      processed. This is a boolean value (1/0). If this is not specified,
      The utility uses 1 - intermediate nodes gets processed.
-->

<ContinueOnFail> </ContinueOnFail>
<!-- See -cof argument -->
<!-- ContinueOnFail specifies whether the creation of the assembly should
      continue if any errors are encountered. This is a boolean value (1/0)
      If this is not specified, the utility uses 1 - continue with creation
      of the assembly.
-->

<DatasetType> </DatasetType> <!-- See -dt_type argument -->
<!-- The dataset type used for attaching the resulting assembly to the Item
      revision. If this is not specified or the type is not found, the
      utility uses a default value based on the JtAssyType value.
-->

<DatasetRelation> </DatasetRelation> <!-- See -dt_rel argument -->
<!-- The dataset relation used for attaching the dataset to the Item
      revision.
-->

<LogFile> </LogFile>
<!-- See -log argument -->
<!-- LogFile identifies the complete path and the file name for the log
      file the utility generates.
-->

<UpdateType> </UpdateType>
<!-- See -assy_update argument -->
<!-- UpdateType determines how to attach the dataset to the Item revision.
      If this is not specified, the utility replaces the named reference,
      if one exists, with the dataset.

```



```

-->

<TargetLoc></TargetLoc>
<!-- TargetLoc specifies the path to the location on your operating system
      where the utility saves the new JT assembly file.-->

<CreateDataset> </CreateDataset>
<!-- See -de argument -->
<!-- The valid values for CreateDataset are:
      n - creates a new dataset every time.
      e - if the dataset already exists it does not revise it. If the dataset
          does not exist it is create.
      a - if the dataset already exists it revises it. If the dataset does not
          exist it is created.
      r - revises the dataset if it already exists. If the dataset does not
          exist an error is thrown.
      If this is not specified, the utility does not create a dataset.
-->

<NamedRef> </NamedRef>
<!-- See -dt_ref argument -->
<!-- The named reference used for uploading the assembly file to the dataset.
      If this is not specified, the utility uses the default named reference
      attached to the dataset type.
-->

</ParamSet> <!-- End of JT file parameters -->
: <!-- Additional ParamSet elements for each set of JT file parameters are here in
-->
</ParamBatch> <!-- End required root element -->

```

See **Examples** for a sample XML file listing.

Note:

If an input XML file is specified, all other command line arguments (with the exception of the log on **(-u, -p or -pf, and -g** arguments) are ignored.

-log

Specifies the full path file name of the log file that stores PLM XML and load library errors.

-rev

Specifies the revision ID of the top line.

If you do not include the **-sc** argument, this argument is required.

-rev_rule

Specifies the revision rule for configuring the assembly.

-var_rule

Specifies the variant rule for configuring the assembly.

-tmode

Specifies the name of the transfer mode. The transfer mode is not stored in the system but is created at run time.

If you do not include this argument, the utility uses the default **JTDataExportDefault** transfer mode.

-jt_type

Specifies the type of assembly to be created. Valid values are:

1

A monolithic JT file is a single file containing all the graphics for a specific assembly.

2

An assembly JT file is an assembly file that contains pointers to other JT files, each representing a component in an assembly.

3

A CoJT file is for use with Tecnomatix eMServer only.

If you do not specify this argument, the utility uses **1** (Monolithic JT).

-pin

Indicates whether to process intermediate nodes. Intermediate nodes are any nodes that are not leaf nodes. Valid values are:

0

Do not process intermediate nodes.

1

Process intermediate nodes.

If you do not specify this argument, the utility uses **1** (process intermediate nodes).

-cof

Indicates whether to terminate assembly creation upon encountering errors. Valid values are:

0

Stop process when encountering an error.

1

Continue processing when encountering an error.

If you do not specify this argument, the utility uses **1** (continue processing).

-dt_type

Specifies dataset type used for attaching the resulting assembly to the item revision.

-dt_rel

Specifies the dataset relation used for attaching the dataset to the item revision.

-dt_ref

Specifies the named reference used for uploading the assembly file to the dataset. If no named reference is given, the default named reference attached to the dataset type is used.

-de

Specifies the behavior of dataset creation.

If you do not enter a value for this parameter, Teamcenter creates a JT file without an associated dataset. Valid value are:

n

Creates a new dataset.

e

Does not revise an existing dataset. If the dataset does not exist, it creates the dataset.

a

Revises an existing dataset. If the dataset does not exist, it creates the dataset.

r

Revises an existing dataset. If the dataset does not exist, it generates an error.

-assy_update

Specifies how the assembly JT is added to the dataset.

create

Assembly file is uploaded if no named reference is found in the dataset.

replace

Assembly file is always created and replaces the current named reference, if it exists.

-skip_root_dset_with_name

Prevents the utility from exporting the dataset of this name that is associated to the root node by an **IMAN_Rendering** relation.

-skip_remote_lines

Prevents the utility from exporting BOM lines that are not loadable.

-Debug

Turns on the debug mode that creates additional debug information files in the log file location.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- The following example is for an input XML file (**robots.xml**) that uses the **StructureContext** element and therefore does not require the **ItemId**, **RevId**, **RevisionRule**, **VariantRule** or **TransferMode** elements. The default **Latest Working** revision rule and **JTDataExportDefault** transfer mode are used. The default dataset type for a monolithic JT assembly is used to attach the dataset to the JT assembly because the **DatasetType** element is not specified:

```
<!-- assy_jt_creator utility robots.xml input file -->
<ParamBatch>
  <ParamSet id="001">
    <StructureContext>CompoundTools</StructureContext>
    <ItemId />
    <RevId />
    <RevisionRule />
    <VariantRule />
    <TransferMode />
    <JtAssyType>1</JtAssyType>
    <ProcessIntermediateNode>1</ProcessIntermediateNode>
    <ContinueOnFail>1</ContinueOnFail>
    <DatasetType />
    <DatasetRelation>IMAN_reference</DatasetRelation>
    <LogFile>c:\temp\monolithicJTUtility\logfile.log</LogFile>
    <UpdateType>create</UpdateType><CreateDataset></CreateDataset>
    <TargetLoc>c:\temp\monolithicJTUtility</TargetLoc>
    <CreateDataSet>n</CreateDataSet>
    <NamedRef>JTPart</NamedRef>
  </ParamSet>
  <!-- uncomment this section to add additional JT file parameters
  <ParamSet id="002">
    :
  </ParamSet>
  -->
</ParamBatch>
```

Use the following command to create the JT assembly using the **robots.xml** file:

```
assy_jt_creator -u=Tc-admin-user -p=password -g=group -i=robots.xml
```

- The following example shows the same command line version creating the same monolithic JT assembly shown in the previous XML file example. However, the command line version does not continue to create the assembly if an error is encountered during the process, does not revise the dataset if one exists, and uses a custom transfer mode. A debug file is also created in the log directory. The entire command must be entered on a single line.

```
assy_jt_creator -u=Tc-admin-user -p=password -g=group -sc=Robots
-jt_type=1 -tar_loc=c:\temp\monolithicJTUtility
-tmode=JTDataExportCustomTM -pin=1 -cof=0
-dt_rel=IMAN_reference -assy_update=create
-de= -dt_ref=JTPart
-debug
```

cc_writer

Creates a PLM XML cache file for a process structure contained in a specified collaboration context object. The PLM XML cache file can be then loaded to Teamcenter lifecycle visualization mockup from within Teamcenter.

SYNTAX

```
cc_writer -cc_name=cc-name
| -cc_uid=uid -sc_name= structure-context-name
| -sc_type=structure-context-type -output_file=file
[-ua=pref:preferenceName,target:targetName1,key:keyName1,prop:propName1
,...,target:targetNameN,key:keyNameN,prop:propNameN] [-h]
```

ARGUMENTS

-cc_name

Specifies the name of the collaboration context object. Either this argument or the **-cc_uid** argument must be specified.

Note:

The collaboration context name is not assumed to be unique. If more than one object is found, the utility returns an error message.

-cc_uid

Specifies the UID of the collaboration context object in Teamcenter. This argument is used in place of the **-cc_name** argument in situations where identifying the collaboration context object by name only is not sufficient because the collaboration context name is not unique in Teamcenter. A query must be defined to obtain the UID of a given collaboration context object.

-sc_name

Specifies the name of the structure context to export. This argument is optional. By default, the composition in the collaboration context object is exported. This argument is used only to export non-composition structure context.

-sc_type

Specifies the type of structure context to export, for example, **MEProcessContext**. This argument is optional.

-output_file

Specifies the name of the file to be created and associated with the dataset. The file is created in the current folder if a path is not specified or in the target directory if a path is specified.

-ua

Defines user attributes. The syntax is similar to the **bomwriter -ua** option.

- **pref:***preference-name-used-for-this-utility*

For example, **pref: CC_ExtraPLMXMLInstanceAttributes** picks the value of **"CC_ExtraPLMXMLInstanceAttributes"** as the user attribute.

- **target:***the-element-in-the-PLM XML-under-which-the- property-is-added*

Valid values are **Part** and **Occurrence**. If **Part**, the property is added to **ProductRevisionView**. If **Occurrence**, the property is added to **Occurrence**.

- **key:***name-of-property-in-PLM XML-file*
- **prop:***name-of-the-property-in-Teamcenter*

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- To create the **cc_process_cache.plmxml** PLM XML file for the process structure found in the structure context of the **cc_process_cache** collaboration context object, enter the following command on a single line:

```
cc_writer -u=Tc-admin-user -p=password -g=group -cc_name=cc_process_cache
-sc_type= MEProcessContext -output_file=cc_process_cache.plmxml
-ua=target:Part, key:Description,prop:bl_item_object_desc,
target:Occurrence, key:FNA, prop:Usage_FNA
```

The PLM XML file contains additional attributes exported as user data. The **Description** attribute is added to the product revision element and the **FNA** attribute is added to the occurrence element. Those attributes can then be presented with their values in Teamcenter lifecycle visualization mockup.

- To create the **va_test.xml** PLM XML file for the process structure found in the structure context *process* of the collaboration context object **va_test**, enter the following command on a single line:

```
cc_writer -u=Tc-admin-user -p=password -g=group -cc_name=va_test
-output_file=va_test.xml -ua=pref:CC_ExtraPLMXMLInstanceAttributes,
target:Occurrence, key:test_name, prop:CompoundName
```

The PLM XML file contains additional attributes exported as user data. The user attribute is the combination of what is specified in the **CC_ExtraPLMXMLInstanceAttributes** preference and the **target:Occurrence,key:test_name,prop:CompoundName** attributes. Those attributes can then be presented with their values in Teamcenter lifecycle visualization mockup.

gcs_import

Imports GCS connection types and connection point definitions from an XML file.

For more information about using the guided component search, see *Resource Manager on Rich Client*.

SYNTAX

```
gcs_import [-u=user-id {-p=password | -pf=password-file } -g=group]  
-xml_file=xml_file_name -import_mode=ignore | overwrite [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-xml_file

Specifies the name of an XML file containing connection types and connection point definitions. The syntax of this file is explained in the *EXAMPLES* section.

-import_mode

Specifies how existing data is handled by the import. Valid values are:

ignore	The existing data is skipped by the import.
overwrite	The existing data is replaced by the contents of the XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To import the connection type and connection point definition found in an XML file named **GCS_data.xml**, enter the following command on a single line:

```
GCS_import -u=user-id -p=password -g=group -xml_file=GCS_data.xml
-import_mode=ignore
```

The **GCS_data.xml** file must have the following format:

```
<GCS_import>
<ConnectionType>
  <Name>CT_Cylinder</Name>
  <Attribute>
    <ID>-40032</ID>
    <ComparisonCriterion><=;</ComparisonCriterion>
  </Attribute>
  <Attribute>
    <ID>-40033</ID>
    <ComparisonCriterion>>=</ComparisonCriterion>
  </Attribute>
</ConnectionType>
<ConnectionType>
  <Name>CT_Square</Name>
  <Attribute>
```

```

        <ID>-40127</ID>
        <ComparisonCriterion>=</ComparisonCriterion>
    </Attribute>
</ConnectionType>
<ConnectionType>
    <Name>CT_Insert</Name>
    <Attribute>
        <ID>-40920</ID>
        <ComparisonCriterion>=</ComparisonCriterion>
    </Attribute>
</ConnectionType>
<ConnectionPointDefinition>
    <ClassID>TC_MILL_10_10_100</ClassID>
    <ConnectionTypeName>CT_Cylinder</ConnectionTypeName>
    <Index>1</Index>
    <Quantity>1</Quantity>
    <Direction>Upwards</Direction>
    <Shape>Plug</Shape>
    <Attribute>
        <ID>-40032</ID>
        <Mapping>#-40235</Mapping>
    </Attribute>
    <Attribute>
        <ID>-40033</ID>
        <Mapping>#-40235</Mapping>
    </Attribute>
</ConnectionPointDefinition>
<ConnectionPointDefinition>
    <ClassID>TC_DRILL_12_10_100</ClassID>
    <ConnectionTypeName>CT_Cylinder</ConnectionTypeName>
    <Index>1</Index>
    <Quantity>1</Quantity>
    <Direction>Upwards</Direction>
    <Shape>Plug</Shape>
    <Attribute>
        <ID>-40032</ID>
        <Mapping>#-40235</Mapping>
    </Attribute>
    <Attribute>
        <ID>-40033</ID>
        <Mapping>#-40235</Mapping>
    </Attribute>
</ConnectionPointDefinition>
<ConnectionPointDefinition>
    <ClassID>TC_HOLDER_20_00_190</ClassID>
    <ConnectionTypeName>CT_Cylinder</ConnectionTypeName>
    <Index>1</Index>
    <Quantity>1</Quantity>
    <Direction>Upwards</Direction>

```

```

    <Shape>Socket</Shape>
    <Attribute>
      <ID>-40032</ID>
      <Mapping>#-40032</Mapping>
    </Attribute>
    <Attribute>
      <ID>-40033</ID>
      <Mapping>#-40033</Mapping>
    </Attribute>
  </ConnectionPointDefinition>
  <ConnectionPointDefinition>
    <ClassID>TC_HOLDER_10_00_110</ClassID>
    <ConnectionTypeName>CT_Square</ConnectionTypeName>
    <Index>1</Index>
    <Quantity>1</Quantity>
    <Direction>Upwards</Direction>
    <Shape>Socket</Shape>
    <Attribute>
      <ID>-40127</ID>
      <Mapping>#-40127</Mapping>
    </Attribute>
  </ConnectionPointDefinition>
  <ConnectionPointDefinition>
    <ClassID>TC_TURN_10_10_100</ClassID>
    <ConnectionTypeName>CT_Square</ConnectionTypeName>
    <Index>1</Index>
    <Quantity>1</Quantity>
    <Direction>Upwards</Direction>
    <Shape>Plug</Shape>
    <Attribute>
      <ID>-40127</ID>
      <Mapping>#-40238</Mapping>
    </Attribute>
  </ConnectionPointDefinition>
  <ConnectionPointDefinition>
    <ClassID>TC_TURN_10_10_100</ClassID>
    <ConnectionTypeName>CT_Insert</ConnectionTypeName>
    <Index>2</Index>
    <Quantity>1</Quantity>
    <Direction>Downwards</Direction>
    <Shape>Socket</Shape>
    <Attribute>
      <ID>-40920</ID>
      <Mapping>#-40920</Mapping>
    </Attribute>
  </ConnectionPointDefinition>
  <ConnectionPointDefinition>
    <ClassID>TC_INSERT_10_00_110</ClassID>
    <ConnectionTypeName>CT_Insert</ConnectionTypeName>

```

```
<Index>1</Index>
<Quantity>1</Quantity>
<Direction>Upwards</Direction>
<Shape>Plug</Shape>
<Attribute>
  <ID>-40920</ID>
  <Mapping>#-40920</Mapping>
</Attribute>
</ConnectionPointDefinition>
</GCS_import>
```

import_nxcam_post_files

Imports CAM post and CSE driver data from an operating system-based file system into the Teamcenter database. For example, you can use this utility to import the files currently being used in the NX environment to the Teamcenter database. In this situation, run this utility only once to initially load existing supporting files into the Teamcenter database. Once the correct NX customer default is set, the files are stored correctly during the sessions.

SYNTAX

```
import_nxcam_post_files [-u=user-id {-p=password | -pf=password-file} -g=group]
-action=ignore | overwrite | new_revision
[-import_dir=import-directory-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

Specifies the actions performed during the import.

- **ignore**

Does not import objects that already exist in the database.

- **overwrite**

Overwrites objects that already exist in the database.

- **new_revision**

Creates a new revisions of the objects that already exists in the database. This is the default setting.

-import_dir

Specifies the directory containing the CAM post and CSE driver data to be imported. If this argument is not used, one of the environment variables listed in the *Environment* section must be set.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

If the **-import_dir** argument is not specified, one of the following environment variables must be set before running this utility:

- **UGII_CAM_POST_DIR**

Imports global postprocessor files and/or posts and ISV drivers for generic machines.

- **UGII_CAM_USER_DEF_EVENT_DIR**

Imports user-defined event files.

- **UGII_CAM_SHOP_DOC_DIR**

Imports shop documentation files.

- **UGII_CAM_LIBRARY_INSTALLED_MACHINES_DIR**

Imports the post/CSE files of installed machines.

For additional information on these environment variables, see the NX documentation.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To import machine post and CSE data from **d:\import_dir** directory using the default action, enter the following command on a single line:

```
%TC_ROOT%\bin\import_nxcam_post_files -u=Tc-admin-user -p=password -g=group
-import_dir=d:\import_dir
```

To import machine post and CSE data from the **d:\mach\resource** directory using the **-overwrite** action, enter the following command on a single line:

```
%TC_ROOT%\bin\import_nxcam_post_files -u=Tc-admin-user -p=password -g=group
-action=overwrite -import_dir=d:\mach\resource
```

To import only global postprocessor files from the **d:\mach\resource\postprocessor** directory to the database using the default action, enter the following two commands.

```
set UGII_CAM_POST_DIR= d:\mach\resource\postprocessor
%TC_ROOT%\bin\import_nxcam_post_files -u=Tc-admin-user -p=password -g=group
-import_dir=d:\import_dir
```

import_step_part21_files

Imports vendor product data (tool components) from a vendor catalog to the **Vendor Catalogs** branch of the classification hierarchy. A user can then map these components with data to a customer branch of the hierarchy and use them to create tool assemblies in Resource Manager for use in NX CAM.

For more information about importing vendor catalogs, see *Resource Manager on Rich Client*.

SYNTAX

```
import_step_part21_files [-u=user-id {-p=password | -pf=password-file} -g=group]
{-import_dir=tool-catalog-import-directory-name|-file=p21-file-with-directory-name}
-map_file=mapping-file-name
-assortment_file=assortment-file-name
[-All]
[-classID]
[-dryrun]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-import_dir

Specifies the directory containing the **p21** files.

-file

Specifies the full path and file name for the **p21** files used to import the vendor catalog.

Teamcenter ignores this parameter if **-import_dir** is set.

-map_file

Specifies the name of a text file that contains the mapping definitions from the **p21** vendor class attributes to the Teamcenter attributes found in the classes in the **Vendor Catalogs** hierarchy in the classification hierarchy. This file is delivered with the vendor catalog.

-assortment_file

Specifies the path and name of a text file that contains a list of tool component data. This file is delivered with the vendor catalog.

-All

Imports all components listed in the assortment file.

-classID

Specifies the class in the hierarchy that acts as a starting point for the import. Teamcenter imports product data (tool component data) from this class downward in the classification hierarchy.

-dryrun

Tests the import by verifying the existence of tool classes and attributes.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To import one **p21** file:

```
import_step_part21_files -u=Tc-admin-user -p=password -g=group
-file=D:\toolItems\5726990_20120829d.p21
-map_file=D:\toolItems\STEP_p21_Mapping_SA__V09.txt
-assortment_file=D:\toolItems\GTC_P21_Assortment_Example_V9_work.txt
```

- To import all **p21** files under one directory:

```
import_step_part21_files -u=Tc-admin-user -p=password -g=group
-import_dir=D:\toolItems
-map_file= D:\toolItems\STEP_p21_Mapping_SA__V09.txt
-assortment_file= D:\toolItems\GTC_P21_Assortment_Example_V9_work.txt
```

- To perform a dry run on one **p21** file:

```
import_step_part21_files -u=Tc-admin-user -p=password -g=group
-file=D:\workdir\p21_files\5726990_20120829d.p21
-map_file= D:\toolItems\STEP_p21_Mapping_SA__V09.txt
-assortment_file= D:\toolItems\GTC_P21_Assortment_Example_V9_work.txt
-dryrun
```

- To import all files listed in the assortment file:

```
import_step_part21_files -u=Tc-admin-user -p=password -g=group
-import_dir=D:\toolItems
-map_file= D:\toolItems\STEP_p21_Mapping_SA__V09.txt
-assortment_file= D:\toolItems\GTC_P21_Assortment_Example_V9_work.txt
-All
```

- To import all files within one class ID in the assortment file:

```
import_step_part21_files -u=Tc-admin-user -p=password -g=group
-import_dir=D:\toolItems
-map_file= D:\toolItems\STEP_p21_Mapping_SA__V09.txt
-assortment_file= D:\toolItems\GTC_P21_Assortment_Example_V9_work.txt
-classID=ADPRS_MSKG51
```

install_mes_templates

Creates a dataset containing all files and relations necessary to generate a 3D PDF report using the Generate 3D PDF Report wizard. As input, this utility requires an XML file that contains the paths to the required files as well as a variety of parameters.

SYNTAX

```
install_mes_templates -u=user-id {-p=encrypted-password | -pf=password-file} [-g=group]
[-type=optional-dataset-type]
[-desc=optional-dataset-description]
-file=qualified-path-to-XML-input-file
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password. This password must be encrypted

You can obtain an encrypted password by typing:

```
ipa_b_executer -p=plain-text-password -encrypt
```

Teamcenter returns the encrypted password as output.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-type

Specifies the dataset type. The default value is **Mes0PDFReportTemplate**.

-desc

Specifies a dataset description.

-file

Specifies the path to the XML file that is used as input for this utility.

For more information, see *XML file syntax*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

XML FILE SYNTAX

The following is a sample XML file required as input for this utility.

```
<?xml version="1.0" encoding="UTF-8"?>
<ME_3D_PDF_ImportData>
  <Templates>
    <TemplateInput name="my new 3DPDF Work instructions">
      <PDF file=".\\Templates\\TC mode\\3D\\my_ME_TC3D_PDF_Template.pdf" type="binary"/>
      <Thumbnail file=".\\Templates\\TC mode\\3D\\my_ME_TC3D_PDF_TemplatePreview.jpg"
        type="binary"/>
      <Stylesheet file=".\\Templates\\TC mode\\3D\\my_ME_TC3D_PDF_StyleSheet.xml" type="text"/>
      <DynamicJS file=".\\Templates\\TC mode\\3D\\my_ME_TC3D_PDF_JavaScript.js" type="text"/>
      <TM name="ME_Gen3D_PDF_ExportTM" file=".\\Templates\\my_ME_Gen3D_PDF_ExportTM.xml"
        overwrite="false"/>
      <Contain2D val="False"/>
      <Contain3D val="True"/>
      <ContainSimulation val="False"/>
      <DataSource val="TC"/>
    </TemplateInput>
  </Templates>
</ME_3D_PDF_ImportData>
```

```

<Override val="True"/>
<Specification val="SHOW_DATASELECTION_PAGE=STEPS_DEF_OPTIONS"/>
</TemplateInput>
</Templates>
</ME_3D_PDF_ImportData>

```

You must modify the following parameters:

ME_3D_PDF_ImportData

Identifies the beginning and end of the XML file.

Templates

Lists all templates.

TemplateInput

Specifies a new template that you must add to the **ME3DPDFAllowedReportTemplates** preference. Each **TemplateInput** element contains all the details for a specific template. You can include multiple **TemplateInput** elements in one **ME_3D_PDF_ImportData** file.

The **TemplateInput** element can have the following elements:

PDF file	Specifies the path to the PDF file that you modify with Adobe LiveCycle Designer.
Thumbnail file	Specifies the path to the JPEG thumbnail file that is used to display the new report template in the list of available templates in the Generate 3D PDF Report wizard.
Stylesheet file	Specifies the path to the XSL style sheet that you can use to modify the format of the report and to add new properties.
DynamicJS file	Specifies the path to the JavaScript file that you can use to modify various control behaviors.
TM name	Specifies the path to the transfer mode that determines what data is included in the report. Generally, you need to modify this file to reflect your data model.
Contain2D val	Specifies whether the report contains 2D data.
Contain3D val	Specifies whether the report contains 3D data.
ContainSimulation val	Specifies whether the report contains simulation data originating from Process Simulate.
DataSource val	Specifies whether the report contains Teamcenter (TC) or Process Simulate (PS) data.
Override val	Specifies whether the report is overwritten each time you run the wizard or if it creates a new report each time.
Specification val	Specifies various options available in the Create 3D PDF Report wizard. Valid values are:

SHOW_DATASELECTION_PAGE=STEPS_DEF_OPTIONS

Specifies that the wizard displays the data selection page.

PRINT_VERSION=True

Specifies that a print version of the report is created along with the interactive template.

MULTI_PAGE=True

Specifies whether the report contains multiple pages. This is applicable to 2D PDF reports only.

PRC_SIMULATION=True

Specifies whether the template is used for **PRC** or **U3D** simulation data. If **ContainSimulation val=True** and **PRC_SIMULATION** is not set or set to **False**, the template supports only **U3D** simulations.

If **ContainSimulation val=True** and **PRC_SIMULATION=True**, the template supports only **PRC** simulations.

ipa_b_executer

Generates or updates in-process assemblies and updates filtered in-process assemblies in a manufacturing process structure. The input for this utility is a parameter file in which you specify whether to update either the IPA, the FIPA, or both simultaneously. You cannot create FIPAs with this utility. You can only create FIPAs from within the Manufacturing Process Planner application.

Note:

Depending on the size of the structure, running this utility can be time-consuming.

For more information about in-process assemblies, see *Manufacturing Process Planning — Usage*.

SYNTAX

```
ipa_b_executer -u=user-id {-p=encrypted-password | -pf=password-file} [-g=group
-f=path-to-parameter-file]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password. This password must be encrypted

You can obtain an encrypted password by typing:

```
ipa_b_executer -p=plain-text-password -encrypt
```

Teamcenter returns the encrypted password as output.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the path to the parameter file (**.txt** file) that is used as input for this utility.

For more information, see *Parameter file syntax*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

PARAMETER FILE SYNTAX

The first entry in each line in the parameter file indicates whether an IPA or FIPA is to be updated. Therefore, there are two types of lines in this file:

- The parameter line for an IPA starts with **#Item**
- The parameter line for a FIPA starts with **#Item_FIPA**

Given these two types of lines, three cases can exist:

- If only an **#Item** line is present in the file, Teamcenter updates the IPA only.
- If only an **#Item_FIPA** line is present in the file, Teamcenter updates the FIPA only.

- If both **#Item** and **#Item_FIPA** lines are present in the file, Teamcenter updates both the IPA and the FIPA.

In the line, each parameter must be separated from its value by the % symbol. If there is more than one value for a parameter, each of these values must be separated by a % symbol.

You can use the following parameters to create or update an IPA:

Item

Specifies the top process ID.

Key

Specifies the top process key in the form of **attr=value,attr2=value2....**

Rev

Specifies the top process revision.

Rule

Specifies the configuration rule.

OG

Specifies the type of IPA occurrence group.

Name

Specifies the name of the IPA.

Occ

Specifies the types of the product occurrences that are included.

Proc

Specifies the types of the processes for which an occurrence group is created.

Use the following parameters to update a filtered IPA:

Item_FIPA

Specifies the process ID of the top process of the BOP structure in which FIPAs are to be updated. This parameter can have only one value.

Rev

Specifies the top process revision. This parameter can have only one value.

Rule

Specifies the revision rule with which the FIPAs are to be updated. This parameter can have only one value.

Parent_FIPA

Specifies the process IDs of the processes for which FIPAs are to be updated. This is mandatory parameter that can have multiple values. In the case of multiple process IDs, the IDs must be separated by a % symbol.

Name

Specifies that only this FIPA in the **Parent_FIPA** process is updated. This is an optional parameter. If no name is specified, then all the FIPAs under the **Parent** process are updated. This parameter can have multiple values.

Prod_ID

Specifies the ID of the top product of the product structure from which items are consumed into the process structure. This parameter can have only one value.

Prod_Rev

Specifies the revision of the top product line. This parameter can have only one value.

Prod_Rule

Specifies the revision rule applied on the product structure. This parameter can have only one value.

The following is a sample line for updating FIPAs:

```
#Item_FIPA%Top-process-id##Rev%current-revision-level#
#Rule%current-revision-rule
#Parent%process-id-1[process-id-2.. process-id-n]#Name%fipa-name-1[%fipa-name-2..%fipa-
name-n]
```

EXAMPLES

These examples reference the following structure.

The screenshot shows the Manufacturing Process Planner (MPP) interface. The top bar displays the title 'Manufacturing Process Planner' and the current process '(* 000029/A;1-demoNested)'. Below the title bar, there are tabs for 'Latest Working', 'Date - "Now"', and 'Not Specified'. The main area is divided into two panes. The left pane shows a hierarchical tree structure of the process, starting with '000029/A;1-demoNested (View)' and branching into various sub-processes like 'Proc6_fipa', 'Proc6_assy', 'Proc5_assy', and several 'Proc1' through 'Proc6' sub-processes. The right pane is a table with the following columns: 'Process Structure', 'Item Type', 'Occurrence Type', and 'Bounding Boxes'. The table lists the details for each process in the tree, including item types like 'MEProcess', 'MEProximityResult', 'OccurrenceGroup', and 'Item', and occurrence types like 'MEFilteredIPA', 'MEWorkpiece', and 'MEConsumed'. Bounding box values are also provided for some items.

Process Structure	Item Type	Occurrence Type	Bounding Boxes
000029/A;1-demoNested (View)	MEProcess		
Proc6_fipa	MEProximityResult	MEFilteredIPA	
Proc6_assy	OccurrenceGroup	MEWorkpiece	
Proc5_assy	OccurrenceGroup		
000019/A;1-D6	Item		-0.004762500000000...
000030/A;1-Proc1 (View)	MEProcess		
000014/A;1-D1	Item	MEConsumed	-0.012220138907432...
000031/A;1-Proc2 (View)	MEProcess		
Proc1_fipa	MEProximityResult	MEFilteredIPA	
Proc1_assy	OccurrenceGroup	MEWorkpiece	
000015/A;1-D2	Item	MEConsumed	0.000000000000000, ...
000032/A;1-Proc3 (View)	MEProcess		
Proc2_assy	OccurrenceGroup	MEWorkpiece	
000016/A;1-D3	Item	MEConsumed	0.000000000000000, ...
000033/A;1-Proc4 (View)	MEProcess		
Proc3_assy	OccurrenceGroup	MEWorkpiece	
000017/A;1-D4	Item	MEConsumed	-0.000793750000000...
000034/A;1-Proc5 (View)	MEProcess		
Proc4_fipa	MEProximityResult	MEFilteredIPA	
Proc4_assy	OccurrenceGroup	MEWorkpiece	
000018/A;1-D5	Item	MEConsumed	-0.004165600115060...
000035/A;1-Proc6 (View)	MEProcess		
Proc5_fipa	MEProximityResult	MEFilteredIPA	
Proc5_assy	OccurrenceGroup	MEWorkpiece	
000019/A;1-D6	Item	MEConsumed	-0.004762500000000...

- To update only the IPA:

```
#Item%000029##Rev%A##Rule%Latest Working##OG%OccurrenceGroup#
#Name%IPA -Jun 30, 2011 12:01:21 PM##Occ%MEConsumed#
#Proc%MEProcess#
```

The file does not contain a line for the FIPA.

- To update only the FIPA for process **Proc2**:

```
#Item_FIPA%000029##Rev%A##Rule%Latest Working##Parent_FIPA%000031#
#Name%Proc2_fipa##Prod_ID%000013##Prod_Rev%A##Prod_Rule#
#Latest Working#
```

The file does not contain a line for the IPA.

- To update the FIPAs for multiple processes:

```
#Item_FIPA%000029##Rev%A##Rule%Latest Working#
#Parent_FIPA %000031%000034%000035#
#Name%Proc2_fipa%Proc5_fipa%Proc6_fipa#
#Prod_ID%000013##Prod_Rev%A##Prod_Rule##Latest Working#
```

The file does not contain a line for the IPAs.

- To update the FIPA for the top process:

```
#Item_FIPA%000029##Rev%A##Rule%Latest Working##Parent_FIPA%000029#
#Name%demoNested_fipa##Prod_ID%000013##Prod_Rev%A##Prod_Rule#
#Latest Working#
```

The file does not contain a line for the IPA.

- To update all the FIPAs present in the process structure:

```
#Item_FIPA%000029##Rev%A##Rule%Latest Working##Parent_FIPA%000029#
#Name ##Prod_ID%000013##Prod_Rev%A##Prod_Rule##Latest Working#
```

The file does not contain a line for the IPAs.

Note:

If a value for the **#Name#** parameter is not present, then all the FIPAs under the **Parent_FIPA** process are updated. You must always include the **#Name#** parameter even if it has no value.

- To update the IPA and multiple FIPAs in the process structure:

```
#Item%000029##Rev%A##Rule%Latest Working##OG%OccurrenceGroup#
#Name%IPA -Jun 30, 2011 12:01:21 PM##Occ%MEConsumed##Proc%MEProcess#
#Item_FIPA%000029##Rev%A##Rule%Latest Working##Parent_FIPA
%000031%000034%000035#
#Name%Proc2_fipa%Proc5_fipa%Proc6_fipa##Prod_ID%000013##Prod_Rev%A#
#Prod_Rule##Latest Working#
```

me_create_mbom

Creates or updates a manufacturing bill of materials (MBOM) from an engineering bill of materials (EBOM). The resulting MBOM structure can contain MBOM-specific levels that represent *make* items.

Note:

- The me_create_mbom utility does not support naming rule as user can specify any prefix/suffix.
- Use this utility only to create a new MBOM. If you need to update an MBOM, use an accountability check or an advanced accountability check.

For more information, see *Manufacturing Process Planning — Deployment and Administration*.

SYNTAX

```
me_create_mbom -u=user-id -p={password | -pf=password-file} [-g=group ]
{-ebomroot=item-id
-key=multi-field-key-of-structure-root
-revid=revision-selector
-scname=structure-context-name|-scuid=structure-context-internal-ID
-rev_rule=revision-rule
-mbomroot=item-id
-mkey=multi-field-key-of-structure-root|
-mrevid=revision-selector
-mscname=structure-context-name|-mscuid=structure-context-internal-ID
-mrev_rule=revision-rule
-scopeid=scope-ID|-scopeidincontext=scope-in-context-ID|-scopekey=scope-in-context-key
-depth=level-from-root-structure
-actiononrelease=1|2|3|4
-update
-usemfk
-log|-logrelation|-logobjectuid
-loglevel=1|2|3
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ebomroot

Specifies the item ID of the engineering bill of materials that is used as the basis for the new or updated manufacturing bill of materials. This value is mandatory unless an **-sname**, **-scuid**, or **-key** is specified.

-key

Specifies the key of the top line when multiple attributes are used to form the unique item ID. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility. This value is mandatory unless an **-sname**, **-scuid**, or **-ebomroot** is specified.

For more information, see *BMIDE for Data Model Design*.

If you use this argument, you must also specify a revision rule using the **-revrule** argument.

-revrule

Specifies the revision rule of the EBOM. This argument is mandatory if you specify the **-ebomroot** or **-key** arguments.

-revid

Specifies an additional revision id to be used with **itemid** to pick a specific revision for ebom root.

-sname

Specifies the name of the structure context representing the EBOM structure within a collaboration context object.

You must specify this value if you do not specify a value for the **-scuid**, **-ebomroot**, or **-key** arguments.

-scuid

Specifies the internal ID of the structure context representing the EBOM structure within the collaboration context object.

You must specify this value if you do not specify a value for the **-sname**, **-ebomroot** or **-key** arguments.

-mbomroot

Specifies the item ID of the manufacturing bill of materials that is to be created or updated. This value is mandatory unless an **-mscname**, **-mscuid**, or **-mkey** is specified.

-mkey

Specifies the key of the MBOM top line when multiple attributes are used to form the unique item ID. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility. This value is mandatory unless an **-mscname**, **-scuid**, or **-mbomroot** is specified.

For more information, see *BMIDE for Data Model Design*.

If you use this argument, you must also specify a revision rule using the **-revrule** argument.

-mrevrule

Specifies the revision rule of the MBOM. This argument is only required when you update an MBOM and is mandatory if you specify the **-itemid** or **-key** arguments.

-mrevid

Specifies an additional revision id to be used with **itemid** to pick a specific revision for the existing MBOM root.

-mscname

Specifies the name of the structure context representing the MBOM structure within a collaboration context object.

You must specify this value if you do not specify a value for the **-mbomroot** or **-mkey** arguments.

-mscuid

Specifies the internal ID of the structure context representing the MBOM structure within the collaboration context object.

You must specify this value if you do not specify a value for the **-mbomroot** or **-mkey** arguments.

-scopeid

Optionally specifies the item ID in the EBOM from which to begin the update. You do not have to update an entire MBOM each time you run this utility. You can choose to update only specific nodes.

This argument cannot be used with **-scopeidincontext**.

If no scope argument is specified, Teamcenter begins the traversal at the top line in the manufacturing BOM.

-scopekey

Optionally specifies the key ID in the EBOM from which to begin the update. You do not have to update an entire MBOM each time you run this utility. You can choose to update only specific nodes.

This argument cannot be used with **-scopeidincontext** or **-scopeid**.

If no scope argument is specified, Teamcenter begins the traversal at the top line in the manufacturing BOM.

-scopeidincontext

Optionally specifies the IDIC of the line in the EBOM from which to begin the traversal. This argument is useful if you have multiple instances of the same subassemblies within an assembly. You can specify exactly which instance to update using the in-context ID.

This argument cannot be used with **-scopeid** or **-scopekey**.

If no scope argument is specified, Teamcenter begins the traversal at the top line in the manufacturing BOM.

-depth

Specifies the number of levels to traverse when creating or updating the MBOM. It may not be necessary to traverse to all leaf nodes in a structure.

-actiononrelease

When running an update, determines the action to take if, when trying to mirror a changed EBOM node, the corresponding node in the MBOM is released.

- 1 Skip the EBOM node and make no changes to the MBOM node.
- 2 Revise and modify the MBOM node in the next working revision to contain the changes.
- 3 Update the properties on released item where permissions permit.

4 Update properties on the MBOM and its children.

Note:

There are many situations that may require one or more **-actiononrelease** options depending on your business process, how data is added to Teamcenter, and other considerations.

For example, If your business process involves a design change in which you may either revise an EBOM node or create a new part, option **1** is relevant. If the business process is to always revise an EBOM node for a design change, option **2** is relevant. To migrate data, you should run the utility on the released parts using option **3**.

The default value is **1**.

-update

Updates an existing MBOM with changes in the corresponding EBOM. If you set this, you must specify the **-mbomroot** and **-mrevrule**, or the **-mscname** or **-mscuid**.

-usemfk

If specified, and the only object type present is defined in the **MEMBOM_Mirror_TypePrefixSuffix** preference, Teamcenter uses that type and ebom **item_id** for the MFK value.

-log

Writes additional failures and information to a log file. If specified, log file is written to disk and not attached to any object. You must specify an absolute path to the file.

-loglevel

Controls the level of logging.

- 1** (Default) Log errors only.
- 2** Log errors and warnings.
- 3** Log errors, warnings, and information.

-logrelation

If **-log** is not specified, uses the internal relation type name to attach the log file to the MBOM root item revision. Default value is **IMAN_reference**.

-logobjectuid

The uid of the persistent object to which the log file must be attached. If an invalid uid is supplied, the log is attached to the MBOM root item revision.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

me_fix_idic

This utility stamps ID in context (IDIC) strings in two ways:

- From a stable manufacturing bill of materials (MBOM) structure to an engineering bill of materials (EBOM) structure in which the IDIC strings are somehow deleted.

In this case, the MBOM and EBOM must be linked in the same collaboration context (CC) object. The utility finds matching lines in the MBOM and stamps them to the EBOM by finding an identical **itemid** and a configurable set of properties such as **bl_plmxml_abs_xform** (transform) in the two structures. The **me_fix_idic** utility does not stamp IDIC values to the EBOM if more than one EBOM line matches the equivalence criteria.

- From product or plant structures to process structures for assigned lines copied before you set the **MECopyIdInContextToAssignedLine** preference using the **-createassignids** option.

Note:

When you use the **-createassignids** option, other options specific to MBOM-EBOM stamping are not valid (see **Arguments** below for details).

SYNTAX

me_fix_idic -u=user-ID {-p=password | -pf=password-file} [-g=group

-ccname=collaboration-context-name | -ccuid=collaboration-context-time-stamp-value
 -mbomscname=manufacturing-bill-of-materials-top-line-value |
 -mbomscuid=MBOM-structure-context-unique-identifier
 -ebomscname=engineering-bill-of-materials-top-line-value |
 -ebomscuid=EBOM-structure-context-unique-identifier
 -properties=properties-text-file-name
 -dryrunfile=dry-run-text-file-name
 -log=log-text-file-name
 -scopeid=itemid-for-scope-of-MBOM-if-top-line-is-not-to-be-stamped |
 -scopeidincontext=IDIC-for-scope-of-MBOM-if-top-line-is-not-to-be-stamped
 -ebomscopeid=itemid-for-scope-of-EBOM-if-top-line-is-not-to-be-stamped |
 -ebomscopeidincontext=IDIC-for-scope-of-EBOM-if-top-line-is-not-to-be-stamped
 -cluserule=custom-closure-rule-to-traverse-MBOM-structure
 -createassignids=stamp-IDIC-for-assigned-lines-from-product-or-plant-structures-to-
 process-structure
 -productSCType=product-structure-context-to-use | -plantSCType=plant-structure-
 context-to-use
 -force_idic_gen=force-IDIC-generation-to-EBOM-or-consumed-lines-in-process-structures
 -debug=flag-for-additional-log-information
 -h]

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

-ccname

Specifies the name of the collaboration context to use in the utility.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-ccname

Specifies the name of the collaboration context object to use in the utility.

-ccuid

Specifies the **uid** (unique identifier) value of the collaboration context object that you display in Teamcenter by using the **Print Object** command.

-mbomscname

Specifies the structure context name of the MBOM object to use in the utility. If the name is not unique, use the **mbomscuid**.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-mbomscuid

Specifies the structure context unique identifier to use if the structure context name is not unique. Find the identifier by selecting the structure context object in the MBOM and using the **Print Object** command.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-ebomscname

Specifies the name of the EBOM structure context object to use in the utility. If the name is not unique, use the **ebomscuid**.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-ebomscuid

Specifies the structure context unique identifier to use if the structure context name is not unique. Find the identifier by selecting the structure context object in the EBOM and using the **Print Object** command.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-properties

Specifies a text file defining a BOM line property per line, such as **bl_plmxml_abs_xform** or **bl_sequence_no**, which the utility uses to search for a matching property between MBOM and EBOM structures. The text file name can be any name, but must sequentially list each BOM line property name, one per line.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-dryrunfile

(Optional) If **dryrunfile=dry.txt**, reports which lines are to be stamped to ensure that the stamped values are valid without actually stamping the target. The *dry.txt* file is created under *tc_menu* folder with details on which lines get stamped.

-log

Creates a log file in the log file location.

-scopeid

(Optional, necessary only if you use scope flow to stamp below the MBOM top line) Specifies the **itemid** of a lower line in the MBOM.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-scopeidincontext

(Optional) If **scopeid** is not sufficient, specifies an IDIC value in the MBOM to determine the scope for traversal in the MBOM.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-ebomscopeid

(Optional, necessary only if you are using scope flow to stamp below the EBOM top line) Specifies the **itemid** of a lower line in the EBOM).

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-ebomscopeidincontext

(Optional) If **ebomscopeid** is not sufficient, specifies an IDIC value in the EBOM to determine the scope for traversal in the EBOM.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-closurerule

(Optional) Specifies a closure rule to be used to traverse the MBOM structure below the scope line. If not specified, the closure rule **MEPrivateTraverseLinesWithIDIC** is created and used.

This argument is not valid if you use **createassignids** to stamp IDIC strings from product or plant structures to process structures for assigned lines.

-createassignids

Specifies that the utility will stamp IDICs for assigned lines from product or plant structures to process structures.

-productSCType

Specifies the product structure context to be used with the **-createassignids** argument.

If not specified, **-createassignids** defaults to **MEProductContext**.

-plantSCType

Specifies the plant structure context to be used with the **-createassignids** argument.

If not specified, **-createassignids** defaults to **MEPlantContext**.

-force_idic_gen

Forces generation of IDICs on the EBOM if you are using the utility to stamp MBOM to EBOM, or on consumed lines if you are using the utility to stamp from product or plant structures to process structures.

Warning:

Since this option forces IDIC stamping even if values are already present, use it with caution.

-debug

(Optional) Turns on debug mode and creates additional debug information files in the log file location. Values are 1, 2, or 3, with higher values resulting in increasingly verbose logging.

EXAMPLE

- To stamp IDICs from MBOM to EBOM:

```
me_fix_idic -u=Tc_admin -p=password -ccname=TestCC -mbomscname=mbomTop1  
-ebomscname=ebomTop1 -properties=prop.txt -log=log.txt  
-dryrunfile=dry.txt
```

- To stamp IDICs from product or plant structures to process structures:

```
me_fix_idic -u=Tc_admin -p=password -ccname=TestCC -createassignids  
-plantSCType=plantTop1 -log=log.txt  
-dryrunfile=dry.txt
```

Note:

You do not need to explicitly specify the process structure. This is derived from the **ccuid** or **ccname**.

me_populate_plantbop_from_bop

Migrates data from a classic bill of process (BOP) to a plant BOP.

The utility's command syntax and arguments are covered here; however, for an overview on the setup needed before using the utility, see Using the me_populate_plantbop_from_bop migration utility.

Set the environment variable **TC_KEEP_SYSTEM_LOG=true** in the window where you run the utility to save the syslogs generated by utility in the **%Temp%** folder.

SYNTAX

```
me_populate_plantbop_from_bop -u=user-id {-p=password | -pf=password-file} [-g=group]
-classicbopcc= [-classicbopitemid= -classicbopscope=]
-plantbopcc= [-plantbopitemid= -plantbopscope=]
[-stationequivalencecallback -stationequivalencemappingfile=file]
[-contentselectionrule= -contentcopyrule=]
[ -dryrunfile=file -runfile=file]
[ -h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If no value, the user's default group is assumed.

-classicbopcc

Classic BOP Collaboration Context (CC) name or CC UID. The CC is assumed to contain only one process BOP structure, properly configured.

-classicbopitemid

Item ID in case classic BOP CC has multiple root processes.

-classicbopscope

Item ID or UID scope to use within a process structure. No MFK support. If not specified, the CC structure root will be selected.

-plantbopcc

Plant BOP CC name or CC UID. The CC is assumed to contain only one plant BOP structure, properly configured.

-plantbopitemid

Item ID in case plant BOP CC has multiple root processes.

-plantbopscope

Item ID or UID scope to use within a process structure. No MFK support. If not specified, the CC structure root will be selected.

-stationequivalencecallback

Calls **USER_BOPLINE_find_target_stations**. Default if **-stationequivalencemappingfile** and **-stationequivalencecallback** are both not specified.

For information on **USER_BOPLINE_find_target_stations**, see the **Integration Toolkit (ITK) Function Reference** section in the **References for Administrators and Customizers** documentation. (WebKey Username and Password may be required).

-stationequivalencemappingfile

Text file that specifies the process to process station mapping.

It can specify either an Item ID to Item ID mapping or an UID to UID mapping. The keys **ItemIDs** or **ItemUIDs** specify which is used. Example: ItemIDs:000123;000456 or ItemUIDs:lkjsdf123;sdlkjfsd123

If both **-stationequivalencecallback** and **-stationequivalencemappingfile** are specified, then the union of the processes and process stations from both methods are used.

-contentselectionrule

Closure rule to control which BOMLines below the classic BOP Process are copied. If not specified, defaults to the rule specified by preference **MEPasteByRuleClosureRule**.

-contentcopyrule

Cloning rule to use on objects migrated from classic BOP to plant BOP. If not specified, defaults to rule specified by preference **MEPasteByRuleCloningRule**.

-dryrunfile

If specified, will print out the information about the objects that will be migrated between classic BOP and plant BOP. Will not actually Paste anything.

-runfile

If specified, will print out the information about the objects that will be migrated between classic BOP and plant BOP during utility execution. You cannot use **-dryrunfile** and **-runfile** arguments at the same time.

-h

Displays help for this utility.

me_stamp_ids

Traverses a structure according to a closure rule and automatically assigns a value to a specific property based on a recipe determined by the value of the **MEIdGenerationPropertySetting** preference.

For more information, see *Manufacturing Process Planning — Deployment and Administration*.

SYNTAX

```
me_stamp_ids -u=user-id -p={password | -pf=password-file} [-g=group ]
{-itemid=item-id|-key=multi-field-key-of-structure-root|
-scname=structure-context-name|-scuid=structure-context-internal-ID}
-rev_rule=revision-rule
[-dryrunfile=dry-run-file ]
[-scopeid=scope-ID|-scopeidincontext=scope-in-context-ID]
[-closurerule=closure-rule-name]
[-forceupdate=force-update]
[-preference=preference-name]
[-allowduplicates=allow-duplicate-IDICs]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-itemid

Specifies the item ID of the structure root. You must specify this value if you do not specify a value for the **-sname**, **-scuid**, or **-key** arguments. If you use this argument, you must also specify a revision rule using the **-revrule** argument.

-key

Specifies the key of the top line when multiple attributes are used to form the unique item ID. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

If you use this argument, you must also specify a revision rule using the **-revrule** argument.

-sname

Specifies the name of the structure context representing the BOM structure within the collaboration context object.

You must specify this value if you do not specify a value for the **-itemid**, **-scuid**, or **-key** arguments.

-scuid

Specifies the internal ID of the structure context representing the BOM structure within the collaboration context object.

You must specify this value if you do not specify a value for the **-itemid**, **-sname**, or **-key** arguments.

-revrule

Specifies the revision rule. This argument is mandatory if you specify the **-itemid** or **-key** arguments.

-dryrunfile

Prints information to the specified file about the engineering BOM line and the absolute occurrence ID (IDIC) that will be stamped.

-scopeid

Specifies the item ID in the manufacturing BOM from which to begin the traversal. This argument cannot be used with **-scopeidincontext**.

If you do not specify this value, Teamcenter begins the traversal at the top line in the manufacturing BOM.

-scopeidincontext

Specifies the IDIC of the line in the manufacturing BOM from which to begin the traversal. This argument cannot be used with **-scopeid**.

If you do not specify this value, Teamcenter begins the traversal at the top line in the manufacturing BOM.

-cluserule

Specifies the closure rule that determines which lines in the structure Teamcenter stamps when it traverses the manufacturing BOM structure below the scope line.

If you do not specify a closure rule, every line in the structure below the given scope line is stamped.

-forceupdate

Overrides any existing values when stamping the lines.

-preference

Specifies the preference name containing the rules for setting the BOM line property. The default preference is **MEIdGenerationPropertySetting**.

-allowduplicates

Specifies whether the utility checks for duplicate in-context IDs when assigning ID string to a BOM line.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- You can use this utility to create values for the **Usage Address** property based on the constituent properties of item ID and item type.

1. Define the recipe for the property value by setting the **MEIdGenerationPropertySetting** preference to:

```
type:Item,key: bl_usage_address,prop:bl_item_item_id,
prop:bl_item_object_type
```

2. Do one of the following:

- Create the usage address property on each line under the top line:

```
me_stamp_ids -u=your-user-name -p=your-password
-g=your-group -itemid=top-line-ID
-revrule=revision-rule
-dryrunfile=name-of-file-holding-dry-run-information
-forceupdate
```

- Create the usage address on selected lines specified in a closure rule under a scope line determined by the specified IDIC (top level) value. In other words, the utility begins with a line that you specify by IDIC, traverses the structure from the IDIC line downward using the given closure rule, and stamps the resulting lines with the usage address string.

```
me_stamp_ids -u=your-user-name -p=your-password -g=your-group
-itemid=top-line-ID
-revrule=revision-rule
-dryrunfile=name-of-file-holding-dry-run-information
-scopeidincontext=in-context-ID-of-the-line-in-the-structure-
to-use-as-starting-point-for-the-closure-rule-traversal
-closurerule=valid-Teamcenter-closure-rule-
that-filters-the-necessary-lines -forceupdate
```

migrate_vrule_configcontext

Migrates collaboration contexts containing single variant rules to collaboration contexts capable of containing multiple variant rules required, for example, when configuring a 120% BOM. This utility queries the database for all configuration context objects where the **variant_rule** property is not empty. It then copies the **variant_rule** value to **variant_rules[0]** and saves it. The utility generates a log file that lists:

- The number of collaboration context objects found in the database.
- The number of instances that have a variant rule.
- The number of objects that were skipped or do not have a single variant rule.
- The number of instances that failed while converting.

For more information about using the 120% BOM feature, see *Manufacturing Process Planning — Usage*.

SYNTAX

```
migrate_vrule_configcontext -u=user-id {-p=encrypted-password | -pf=password-file} [-g=group]
[-filename=log-file-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

The user needs appropriate write access to the source and target structures in order to create occurrences groups and attach reports.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password. This password must be encrypted.

You can obtain an encrypted password by typing:

```
ipa_b_executer -p=plain-text-password -encrypt
```

Teamcenter returns the encrypted password as output.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-filename

Specifies a user defined file name that holds the log file, for example,
-filename=D:\myFolder\myFile.

If a file name is not specified or remains blank, Teamcenter uses the **C:\temp\migrate_vrule_configcontext_timestamp** file as the default.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

mrm_export_resources

Exports resources from Teamcenter classification. This is useful if you manage your resource and classification data in Teamcenter but run NX in its native mode. In this situation, you can use this utility to export tooling classification and data from Resource Manager so it can be used by NX CAM in native mode.

SYNTAX

```
mrm_export_resources [-u=user-id {-p=password | -pf=password-file} -g=group]
-class=root-class-ID [-locale=locale]
-def_file=definition-file [-class_graphics_dir=directory
[-class_graphics_option=all|changed]]
-dat_file=database-file [-dat_graphics_dir=directory
[-dat_graphics_option=all|changed]] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-class

Exports the classification hierarchy and/or instance data from the specified root class ID. Use the **-def_file** and **-dat_file** arguments to specify whether classification hierarchy, and/or instance data, is exported, respectively.

-def_file

Exports the specified definition file. The file is based on the class hierarchy structure.

Note:

The definition file includes the list of required attributes and key-LOV definitions. It also includes the hierarchical structure of the nested classes.

For each class, the list of available attributes for the **Search** dialog box and the resulting table must be defined. If a definition for the **RSET** attribute set is defined, the attributes specified for the **Search** dialog box is used for the **RSET** attribute set.

For an example of the definition file, see the *Examples* section.

-class_graphics_dir

Exports classification graphics used for the **Search** and the **RSET** dialog boxes to the specified directory.

-class_graphics_option

Specifies whether to export all classification graphics or only those graphics that are modified. Valid values are **all**, which exports all graphics, and **changed**, which exports only graphics that are modified.

-dat_file

Exports the specified ASCII database file. The file includes parameter values for all instances. Each data line also specifies the tool classification.

-dat_graphics_dir

Exports resource graphics to the local file system.

-dat_graphics_option

Specifies whether to export all resource graphics or only those graphics that are modified. Valid values are **all**, which exports all graphics, and **changed**, which exports only graphics that are modified.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To export the definition file for the Manufacturing Tooling Library:

```
mrm_export_resources -u=Tc-admin-user -p=password -g=group -class=TLCUA
-def_file=dbc_mfg_toollib_customer_tlas.def
```

To export the ASCII tool database file for all tool assemblies, including all updated graphic files:

```
mrm_export_resources -u=Tc-admin-user -p=password -g=group -class=TOOL02
-dat_file=tool\metric\tool_database.dat -dat_graphics_dir= tool\
metric\graphics\ -dat_graphics_option=changed
```

A sample definition file:

```
DB_ALIAS DCII
{
  DB_ID      -500118
  DB_ID_TYPE s
  DIALOG_NAME "Cutting diameter Dc"
  RSET_NAME   "Cutting diameter Dc"
}
DB_ALIAS PartType
{
  DB_ID      -3651
  DB_ID_TYPE s
  OPTIONS    "Right" "Left"
  OPTIONS_IDS "R" "L" DIALOG_NAME "Type"
  RSET_NAME  "Type"
}
CLASS TOOL
{
  TYPE QRY
  QUERY "[DB(Type)] == [TOOL01] && [DB(Type)] == [TLCUA]"
  DIALOG libref
  RSET libref
  UI_NAME "Tool"

  CLASS TOOL_MRM
```

```
{
  TYPE QRY
  QUERY "[DB(Type)] == [T00L01]"
  DIALOG libref Holder
  RSET libref Descr MaterialDes Holder
  UI_NAME "MRM Tooling"

  CLASS MILLS
  {
    TYPE QRY
    QUERY "[DB(Type)] == [TAM02]"
    DIALOG libref Diameter Holder
    RSET libref Descr Diameter MaterialDes Holder
    UI_NAME "Milling"
```

mrm_migrate_machining_data

Converts NX feeds and speeds and machining data library ASCII files into Teamcenter SML files and stores the output in a file named **migrated_machining_data.sml** in the directory in which you run the **mrm_migrate_machining_data** utility. After converting, import the SML files into the database using the **smlutility** utility.

SYNTAX

```
mrm_migrate_machining_data [-u=user-id {-p=password | -pf=password-file} -g=group]
-root_dir=NX-feeds-and-speeds-DAT-file-directory [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-root_dir

Points to the directory containing NX feeds and speeds DAT files.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
mrm_migrate_machining_data -u=user-id -p=password -g=dba  
-root_dir=d:\NX9.0\MACH\resource\library\feeds_speeds\ascii\
```

mrm_plmxml_updater

Compares two PLM XML files and their corresponding folders that contain images, and generates a script containing the differences. You can use this utility to create scripts to update the Manufacturing Resource Library's (MRL) class hierarchy according to an exported customer PLM XML file and the PLM XML file found in the MRL kit.

Note:

Running this script can be time-consuming, depending on the number of differences found when comparing the two files.

SYNTAX

```
mrm_plmxml_updater -customer=customer-PLM-XML-file -master=PLM-XML-file-in-MRL-kit
[-configurationfile=configuration-file]
[-outputdirectory=output-directory]
[-v]
[-h]
```

ARGUMENTS

-customer

Specifies the customer PLM XML file. This is the class hierarchy for one MRL module that is exported from the customer's database. The folder with the class images and icons should be in the same directory.

-master

Specifies the master PLM XML file from the MRL kit for the same MRL module that is specified in the **customer** argument. The folder with the class images and icons should be in the same directory.

-configurationfile

Specifies the path and file name of the configuration file that configures specific processing of the utility. This parameter is optional. If not specified, the default settings are used. If this argument is not specified, a default configuration file, **configurationfile.xml**, is created in the directory where the utility is started. The default configuration file can subsequently be modified and used as the input configuration file for the next comparison.

-outputdirectory

Specifies the path to the directory where all output files are stored. This parameter is optional. If not specified, the output files are stored in a directory entitled **UPDATE** located in the directory where the utility is started. If the directory does not exist, the utility creates it automatically.

-v

Provides more output on the currently running operation.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Empty spaces in a path or file name must be surrounded by quotation marks.

EXAMPLES

```
mrm_plmxml_updater -customer=D:\customer\mrl_tools_hierarchy_en.xml
-master=D:\advanced_installations\
resource_management\MRL\ImportFiles\tools\mrl_tools_hierarchy_en.xml
-outputdirectory=D:\OUT\tools
```

With verbose and configuration file set:

```
mrm_plmxml_updater -v -customer=D:\customer\mrl_tools_hierarchy_en.xml
-master=D:\advanced_installations\
resource_management\MRL\ImportFiles\tools\mrl_tools_hierarchy_en.xml
-outputdirectory=D:\OUT\tools
-configurationfile=D:\conf\configurationfile.xml
```

The following update script files are created by the utility:

Update script	Contains
1_KEYLOV_PLMXMLIMPORT.bat	The plmxml_import commands that import the new and updates KeyLOVs
2_ATTRIBUTE_PLMXMLIMPORT.bat	The plmxml_import commands that import the new attributes
3_ATTRIBUTE_UPDATE.bat	The smlutility commands that update the attributes
4_CLASS_PLMXMLIMPORT.bat	The plmxml_import commands that import the new classes
5_CLASS_UPDATE.bat	The smlutility commands that update the classes
6_VIEW_PLMXMLIMPORT.bat	The plmxml_import commands that import the new views
7_VIEW_UPDATE.bat	The smlutility commands that update the views

The numbers of the files indicate the order in which they must be run to update the MRL hierarchy.

In addition to the update scripts, the following directories are generated.

Directory	Contains
ATTRIBUTES_XML	The PLMXML files for the new attributes
CLASSES_XML	The PLMXML files for the new classes
KEYLOVS_XML	The PLMXML files for the new/updated KeyLOVs
VIEWS_XML	The PLMXML files for the new views
INFO_FILES	Information text files with the comparison

For each of the following directory types, six information files are generated in the **INFO_FILES** directory:

- Key-LOVs
- Attributes
- Classes
- Views

For example, for key-LOVs:

File	The file
KEYLOV_LIST_ALL_CUSTOMER.txt	Lists all key-LOVs from the customer PLM XML
KEYLOV_LIST_ALL_MASTER.txt	Lists all key-LOVs from the master PLM XML
KEYLOV_LIST_CUSTOMER.txt	Lists all key-LOVs that are in the customer PLM XML, but not in the master (= customer key-LOVs).
KEYLOV_LIST_NEW.txt	Lists all key-LOVs that are in the master PLM XML but not in the customer (that is, new key-LOVs)
KEYLOV_LIST_MODIFIED.txt	Lists all key-LOVs to be modified during the update
KEYLOV_LIST_INFO.txt	Describes exactly how each key-LOV is modified during the update.

An example of the MRL update configuration file is listed in the *Configuring the update* section of the *Windows Server Installation* guide.

tcexcel_import

Populates structures in Teamcenter based on definitions in a Microsoft Excel spreadsheet. The main purpose of this utility is to import process structures, such as build sequences or process plans, but the utility can also import any type of product, process, plant, and resource information. Additionally, the utility supports:

- Linking between multiple structures
- Assigning relations (consumed, required, work area, and resources)
- Attaching forms and filling out attribute values
- Modifying ownership
- Defining activities (with time information)
- Attaching variant information

The **tcexcel_import** utility is designed for data creation only. You can set the description on items and revisions during creation. Once you create an item, you can only update the item description.

The basic steps to use this utility are:

1. Create an Excel spreadsheet containing the structure that you want to import.
2. Save the spreadsheet as a tab delineated .txt file.
3. Run the **tcexcel_import** utility using the tab delineated .txt file as input.

The utility converts the .txt file to a .pim file and uses the .pim file as input to create the structure in Teamcenter.

This documentation describes the .pim file syntax that is normally created automatically from the .txt file in addition to the parameters for calling the utility.

SYNTAX

```
tcexcel_import [-u=user-id {-p=password | -pf=password-file} -g=group]
-i=input-file1[;input-file2...] [-d=debug-level] [-o={on | off}]
[-t=item-type]
[-s=dummy-status] [-pr={precise | imprecise}]
[-m=marker-file-name] [-f=file-format-help] [-psfile] [-parser_only]
[-delimiter=delimiter-character] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-i

Specifies input files. At least one file name is required. Additional file names are separated with a semicolon (;).

-d

Sets debug level. Default value is 0, no debug messages.

Set this value to 5 or above to generate debug messages. Each level displays progressively more messages. The content of the messages is dependent on the type of structure that is loaded.

-o

Toggles overwrite mode on or off. Default is to overwrite.

-t

Specifies default item type. Default is **item**.

-s

Specifies dummy release status for working revisions. Default is **DUMMY_STATUS**.

-pr

Specifies precise mode. Default is **imprecise**. Set this value to **precise** to create a precise structure.

-m

Specifies marker file name. Default value is **marker.txt**.

-f

Displays a help message describing file format.

-psfile

Specifies if the file specified by the **-i** option is a PIM file.

-parser_only

Specifies to stop after parsing of Excel files with Header4.

-delimiter

Specifies delimiter character. Default is **#**.

Note:

The character used as the **-delimiter** should not be used in the input file. For example, if the delimiter character is **#**, the following line will not process correctly – "SHEET, COP #22 AWG".

-unique_working_revision

Specifies that it is an error to have more than one revision with no status. Only the working revision will be the one with no status.

-align_bv_owner=<1/0>

Specifies that the bv and bvr should have the same owner as the parent item/revision. Default is 1.

-clr

If a prior marker file is present, then delete it before processing.

-add_note

If note types **MEConsumedInfo**, **MERequiredInfo**, **MEWorkareaInfo**, and **MEResourceInfo** are desired

-allow_h4_duplicates

If the same item name appears, the default is to ignore it. Set this option to override that behavior.

-h

Displays help for this utility.

INPUT FILE FORMAT

For information regarding file format and valid data fields, see the Importing structures using the `tcexcel_import` utility section in the Manufacturing Process Planner documentation.

update_stx_elements

Updates the preview files of textual work instructions that contain specific standard text elements. The purpose of this utility is to:

- Systematically find the operation and process revisions that contain textual work instructions that reference a specific standard text element.
- Update the preview files offline rather than singly in rich client.

In a typical case, one or more standard text library elements are revised, reviewed, approved and released. The basic steps to perform the mass update of preview files in relevant textual work instructions are:

1. In My Teamcenter, create a change notice and add the standard text element revisions to the change notice's **Problem Items** folder.
2. Apply the release status to the change notice revision.
3. Create or identify a saved revision rule that will configure the released revision of the standard text elements.
4. Identify or create one or more saved revision rules that will configure the revisions of the operations and processes whose work instructions need to be updated.

You may use more than one rule, such as a latest working rule and a rule for the most recently released and exported revisions.

5. Run the utility to search for processes and operations, specifying as input the change notice ID, the revision rule names, logon credentials, and the **findImpacted** argument.
6. Upon completion of the search, the change item's **Impacted Items** folder contains any process and operation revisions whose work instructions need to be updated.

If necessary, you can review and modify the list.

7. Run the utility again to update the textual work instruction preview files, this time using the **updateImpacted** argument.
8. Once complete, send the updated process and operation revisions in the **Impacted Items** folder to downstream applications for export.

Note:

- The utility updates all of the standard text elements in each textual work instruction according to the stx revision rule. Standard library symbol objects are not supported.
- For the utility to work, the preference **MEWiWorkInstructionMacroTemplate** must be set to **TWIfor_TC1016OOTB_macro_spec_template**. If you use a custom specification template, it must contain all the macros in **MEWiWorkInstructionMacroTemplate**.
- The utility requires a Microsoft Office version compatible with Teamcenter 10.1, as defined in the Hardware and Software Certifications knowledge base article on [Support Center](#), and a Teamcenter Manufacturing Documentation license.
- The logon user must have necessary read and write privileges.

SYNTAX

```
update_stx_elements [-u=user-id {-p=password | -pf=password-file} -g=group]
[-cnid=change-notice-item-id]
[-stxRevRuleName=rev-rule-for-configuring-stx-text-element]
[-structRevRuleNames=comma-delimited-list-of-revision-rule-names-for-
  configuring-process-and-operation-revisions]
{-findImpacted | -updateImpacted}
[-h]
```

ARGUMENTS**-u**

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-cnid

Specifies the change notice ID.

-stxRevRuleName

Specifies the revision rule used to configure the standard text (stx) element.

-structRevRuleNames

Specifies a comma-delimited list of revision rule names for configuring process and operation revisions.

-findImpacted | -updateImpacted

The first argument, **findImpacted**, finds operations or process revisions that have textual work instructions affected by selected standard text elements. The second, **updateImpacted**, updates the textual work instructions attached to those operations or process revisions with the standard text elements.

-h

Displays help for this utility.

upgrade_pv_with_visiblelineinfos utility

This utility lets you upgrade product structure views in a plant bill of materials (BOP) or in a classic BOP to correct problems when you upgrade from earlier Teamcenter releases to 11.6.

Usage

```
upgrade_pv_with_visiblelineinfos -u=user-ID {-p=password | -pf=password-file} -g=group
-log=absolute-path-to-logfilename
-rootitemid=root-item-ID-of-BOP | -key=assembly-top-item-key | -ccname=configuration-context
-name | -mode= [bopwithpv-outputfile=XML-output-file | upgradepv -inputfile=XML-input-file |
pvsneedingmigration -outputfile=XML-file-of-BOP-IDs | upgradeallpvs]
```

Syntax

-log=log-text-file-name. If **logfile** is not specified or blank, c:\temp\upgrade_pv_with_visibleLineInfos_<timestamp> is taken as default.
-debug=y. Starting in debug mode will print detailed errors/warnings on the screen.

-rootitemid=root-item-ID-of-the-classic-BOP-or-plant-BOP-with-product-views-to-be-upgraded.
-key=multi-field-key-of-the-structure-root-that-must-be-specified-if **-rootItemID**-or-**ccname** is-not-specified.
-ccname=configuration-context-that-contains-the-BOP-structure-with-the-product-view.
-mode=bopwithpv. Extracts-all-the-product-views-to-be-upgraded-with-the-structure-names-and-modified-date-reported-in-an-output-XML-file. The-location-of-the-output-XML-file-is-provided-by **-OutputFile**=XML Output File. This-command-is-used-before-you-define-the-mode-as **upgradePV**. -Until-you-run- **upgradePV**-product-views-are-not-upgraded.
-mode=upgradepv. Upgrades-all-product-views-from-the-XML-file-from **-inputFile**=XML Input File. The-ones-which-are-not-upgraded-will-be-found-in-the-log-files.
-mode=pvsneedingmigration. Extracts-all-ItemIDs-for-BOPs-containing-product-views-to-be-upgraded-in-an-XML-file. The-location-of-the-XML-file-is-provided-by **-OutputFile**=XML-Output-File.
-mode=upgradeallpvs. Forces-all-product-views-in-a-database-to-be-upgraded.
-h=help.]

Configuring the Machining Data Library

When you search for machining data in the Manufacturing Resource Library database, such as feeds and speeds, you can configure the searches by activating or deactivating queries that are used to search the data. Modify the **mrl_feeds_speeds_config.txt** configuration file to support your specific requirements.

For example, when you select **Set Machining Data** , enable the following query types and query in the Machining Data Table for combinations of the Library References below. To disable the query, set the item to **0**.

- `smd_query_Machine_Method_PartMaterial_ToolMaterial_ToolDiameter = 1`
- `smd_query_Machine_Method_PartMaterial_ToolMaterial_ToolDiameter_Interpolation = 1`
- `smd_query_Machine_Method_PartMaterial_ToolMaterial = 1`
- `smd_query_Method_PartMaterial_ToolMaterial_ToolDiameter = 1`
- `smd_query_Method_PartMaterial_ToolMaterial_ToolDiameter_Interpolation = 1`
- `smd_query_Method_PartMaterial_ToolMaterial = 1`

Note:

The queries that are activated are executed in the order shown above. If a query is deactivated, it is skipped and the next query in the list is executed.

Two other key benefits with the new configuration file are:

- You can use the methods available previous to Teamcenter 11.5. To reactivate them, set the value of the following entries in the configuration file to **1**.
 - `use_old_functionality_for_machining_data = 1`
 - `use_old_functionality_for_feeds_speeds = 1`
- When no data is found after a query, the user can analyze which parameters were passed to the system by using the report option.
 - `report_level = 1`

You can find the **mrl_feeds_speeds_config.txt** in
`\resource_management\MRL\nxlib\MACH\resource\library\feeds_speeds\inclass.`

Retrieve a structure's Unique Identifier (UID)

UIDs can often be used to identify structures in Teamcenter utilities such as **accountability_check**. Do the following to retrieve a structure's UID.

1. Go to My Teamcenter and open the structure for which you want to retrieve the UID.
2. Choose **Window→Show View→Print Object**.

The UID is displayed in the upper-right corner as a 14-character unique value.

3. Select and copy the UID to a text editor such as Notepad.
4. Do the same for any other structures for which you need UIDs and use the values in your utility command.

10. Service Lifecycle Management utilities

plf_attach

This utility is used to add a part logistics form (PLF) to any items for which the PLF is missing.

In BMIDE, the constant **IsMRONeutralType** must be set to true for the item to be an **MRO Neutral Type**, which is necessary to attach a PLF.

- The items to which the utility is applied are listed in a separate text file.
- You can attach a PLF with this utility to a single item or a list of items.
- If you include a parent in the list of items to add a PLF, Service Lifecycle Management also adds the PLF to the children.
- If any item in the list already has a PLF attached, it is not affected when the utility is executed.

Syntax

```
plf_attach [-h] -u=user_id -p=password  
          | -g=group | -StructureAttributesDetailsFile=item-list-filename
```

Arguments

-u	User name
-p	User password
-g	Group name
-StructureAttributesDetailsFile	Text file containing the item IDs of items to which the PLF is attached. • Do not include tab characters in the text file. • Comments are added by starting a line with a number sign #. • The text file lists item IDs. • Example 1—multifield key (MFK) disabled:

Example command:

```
plf_attach -u=TC-admin-user -p=password -g=dba  
-StructureAttributesDetailsFile=structureDetails.txt
```

Example list of item IDs:

```
item_id=001235
```

```
item_id=001237
```

```
item_id=001238
```

- **Example 2—multifield key (MFK) enabled for Item Type:**

```
item_id=<xxxxxx> <other MFK attributes, if MFK is enabled for Item Type>.
```

Example list of item IDs:

```
item_id=001235 object_name=Part1
```

```
item_id=001235 object_name=Part2
```

```
item_id=001235 object_name=Part3
```

Note:

object_name is an example.



11. Classification utilities

icsutility

Imports the following types of Classification data:

- Class definitions.

This can optionally include the attributes, groups, parent classes, subclasses, or the class hierarchy from the root.

- Any individual attribute or Key-LOV object.

- Resource Manager resource assemblies.

This can optionally include root-level components, intermediate components, bottom-level components, and propagation start points. The hierarchical component positions are maintained.

icsutility also allows you to import class-specific and instance-specific files.

Note:

This utility is primarily used to import legacy data. If you need to do this, contact your Siemens Digital Industries Software representative. To import and export classification data, use PLM XML.

SYNTAX

icsutility[-u=user-id {-p=password | -pf=password-file} -g=group]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mod

Specifies the type of modification: **insert**, **update**, or **revise**.

insert

Creates a new ICO, item, and item revision. If an ICO or item with the corresponding name already exists, nothing is imported.

Attribute values are imported in the new ICO. Class-specific and instance-specific files (for example, **HPGL** files, **GIF** files, **JT** files, and **PRT** files) are attached to the new item revision.

-update

Updates the values in the ICO attached to the latest item revision (or to the item) and the assembly structure in the latest item revision (or item), as well as the class-specific and instance-specific files.

-revise

Creates a new item revision and ICO and imports the attribute values in the new item revision and ICO that classifies the new item revision. Class-specific and instance-specific files are attached to the new item revision.

You can specify an additional argument, **-forceConversion=1** (default value **0**). In this case, the relationship to the item is converted to a relationship from the ICO to the latest item revision.

-dbg

Specifies the debug level. Any positive integer number up to level **7**. **0** turns off debugging mode. The higher the debug level, the more detailed is the trace being output.

-del

Specifies the list of characters used to delimit multiple search directories and file extensions in the following file path and extension options:

-fid

Specifies the name of a folder into which imported items and datasets are placed. This folder is placed in the **Newstuff** folder.

-sfn

Specifies the name of the SML file to be processed. Do not include the **.sml** file extension.

-sfp

Specifies the path of the directory containing the **.sml** file. Paths must be terminated using the platform-specific path delimiter, as follows:

\ Windows delimiter

/ Linux delimiter

-cfp

Specifies the directory path (or paths) containing the class-related files to be imported, such as class image files.

-cfe

Specifies the file extension of the class-related files to be imported. Accepts multiple file extensions separated by the delimiter set using the **-del** argument.

-ofp

Specifies the directory path (or paths) of the object-related files to be imported.

-ofe

Specifies the file extension of the object-related files to be imported. Accepts multiple file extensions separated by the delimiter set using the **-del** argument.

-ffp

Specifies the directory path containing the file types configuration file to be used.

-ffn

Specifies the name of the file types configuration file to be used, including the file extension.

-clr

If used, specifies that item revisions are classified when importing data. If not used, items are classified upon import.

When you specify the **-clr** argument and try to import an ICO that is already in the database and the item, but not the item revision, was classified, an error message is output when **insert** mode is used. When **update** mode is used, the relationship to the item is converted to a relationship from the ICO to the latest item revision.

-cit

Creates items of the specified item type, rather than the standard items.

ENVIRONMENT

This utility must be run in the Teamcenter shell environment.

FILES

As specified in *Log files produced by Teamcenter* and the **filetypeDefaults.txt** configuration file.

The entries in the **filetypeDefaults.txt** configuration file map the extensions of all imported files to specific Teamcenter data structures: GRM relationship type, dataset type, named reference, optional default tool, and optional subdirectory name. The mapping of file extensions to specific Teamcenter data structures can be configured per import directory/subdirectory combination.

RESTRICTIONS

You must have the corresponding privileges (create/modify) to start the import process. If you do not have the privileges, the data is not imported and a message is written to the log file.

Only the initial population of an empty Resource Manager database is fully supported.

EXAMPLES

To import the **C:\import\sml\test.sml** file using the file type configuration file **C:\import\config\filetypeDefaults.txt** together with GIF class image files residing in the **C:\import\class** directory and object-related JT, GIF, and Word files residing in the **C:\import\object** directory, enter the following command on a single line:

```
icsutility -u=smith -p=secret -g=admin
  -mod=update -dbg=1 -del=, -fid=import01 -ffp=C:\import\config\
  -ffn=filetypeDefaults.txt -sfp=C:\import\sml -sfn=test
  -cfp=C:\import\class\ -cfe=gif -ofp=C:\import\object\ -ofe=jt,gif,doc
```

ics_connect

Associates classification objects (ICOs) with workspace objects, based on item ID. You can list multiple workspace objects, or you can create a text file containing a list of item IDs.

For example, when you use the **smlutility** utility to import classification data into Teamcenter, the ICOs are created as specified in the input file, but they are not attached to the items defined in the input file via their workspace object **uid**. Use the **ics_connect** utility to associate all your ICOs (not already connected to any item) to the specified item of the same ID.

Caution:

If you use multifield keys to define item IDs, you cannot use **ics_connect** to connect a standalone ICO to an existing item because multiple items with the same ID exist.

SYNTAX

```
ics_connect [-u=<user-id> {-p=<password> | -pf=<password-file>} -g=<group>} -names=<wso-list>
| -file=<path to file> -key=keyAttr1=keyVal1,keyAttr2=keyVal2,keyAttrN=keyValN | -keyFile=<path to
file>} [-h]
```

EXAMPLES

```
ics_connect -u=Tc-admin-user -p=password -g=group -names=item1
```

```
ics_connect -u=Tc-admin-user -p=password -g=group -names=item2,item3
```

```
ics_connect -u=Tc-admin-user -p=password -g=group -names=item4/A,item5/C
```

```
ics_connect -u=Tc-admin-user -p=password -g=group -file=wso_names.txt
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-names

Specifies the item IDs of the items to be associated with the ICOs of the same ID.

-file

Specifies the name of the file containing the list of item IDs to be associated with the ICOs. Place one item or item revision ID per line.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To associate the ICOs with IDs of **item2** and **item3** with the items with IDs of **item2** and **item3**:

```
ics_connect -u=Tc-admin-user -p=password -g=group -names=item2,item3
```

The system searches for items with the ID **item2** and **item3**, connecting them to ICOs with the same ID, as long as the ICOs are not attached to any other workspace object.

- To associate the ICOs with the **item4/A** and **item5/C** item revisions:

```
ics_connect -u=Tc-admin-user -p=password -g=group -names=item4/A,item5/C
```

The system searches for item revisions with the ID **item4/A** and **item5/C**, connecting them to ICOs with the same ID, as long as the ICOs are not attached to any other workspace object.

- To associate the ICOs with the item IDs contained in the **wso_names.txt** file:

```
ics_connect -u=Tc-admin-user -p=password -g=group -file=wso_names.txt
```

The file must contain item and item revision IDs, one per line.

- To connect Workspace objects with MFK IDs use the following:

Note:

To find the key of an object, use the **get_key_string** utility.

```
ics_connect -u=Tc-admin-user -p=password -g=dba  
-key=item_id=000039,object_name=ABC
```

- To connect the revision use:

Note:

The revision has to be last entry.

```
ics_connect -u=Tc-admin-user -p=password -g=dba  
-key=item_id=000039,object_name=ABC,rev_id=A
```

- When using the **-keyFile** option, use the following format:

```
item_id=export_001
```

```
item_id=M2Item1_001,object_name=M2Item_name1,object_type=M2Item1
```

```
item_id=M2Item1_002,object_name=M2Item_name2,object_type=M2Item2,rev_id=B
```

lbrmanager

Performs various library management functions. Using a multitude of arguments, you can create, update, and delete data, show data, publish and retract it, as well as search for various library elements.

To use the **lbrmanager** utility, you must install the **Teamcenter Classification Collector** feature in your Teamcenter configuration.

SYNTAX

lbrmanager -u=*user-name* {-p=*password* | -pf=*password-file*} -g=*group-name*

[**-create** | **-delete** | **-show** | **-find** | **-update** | **-revise** | **-copy** | **-share** | **-unshare** | **-evaluate** |
-publish | **-retract** | **-instantiate** | **-search** | **-clone** | **-import** | **-h**]

ARGUMENTS

The **lbrmanager** utility contains many arguments and subarguments whose descriptions and syntax you can find in the help contained within the utility.

- List the utility help in the customary fashion:

```
lbrmanager -h
```

- List the utility help for subarguments as follows:

```
lbrmanager -subargument -h
```

For example:

```
lbrmanager -show -h
```

Running this command lists the help for all the subarguments of the **-show** command, such as the following:

```
Usage: lbrmanager -show -libraries
[-id=<ID> | <MFK of the form: attr1=value, attr2=value...,
  object_type=boName>]
[-norecure : Do not show the Library structure.]
[ -h | -help : This option will print usage details for this operation. ]
```

ENVIRONMENT

This utility must be run in the Teamcenter shell environment.

FILES

As specified in Log files produced by Teamcenter and the **filetypeDefaults.txt** configuration file.

The entries in the **filetypeDefaults.txt** configuration file map the extensions of all imported files to specific Teamcenter data structures: GRM relationship type, dataset type, named reference, optional default tool, and optional subdirectory name. The mapping of file extensions to specific Teamcenter data structures can be configured per import directory/subdirectory combination.

clsutility

Performs administrative tasks when the presentation layer is installed.

SYNTAX

clsutility **-u**=*user-name* {**-p**=*password* | **-pf**=*password-file*} **-g**=*group-name*

[-create | -delete | -find | -ask | -add | -import | -search | -set | -update | -get | -remove |
-describe | -migrate | -h]

ARGUMENTS

The **clsutility** utility contains many arguments and subarguments whose descriptions and syntax you can find in the help contained within the utility.

- List the utility help in the customary manner:

```
clsutility -h
```

- List the utility help for subarguments as follows:

```
clsutility -create -h
```

Running this command lists the help for all the subarguments of the **-create** command, such as the following:

USAGE

```
clsutility
```

```
-u=<username>
```

```
{-p=<password> | -pf=<password_file> }
```

```
-g=<group>
```

```
-output=<output file name>
```

```
=====
```

Description:

Creates default Classification Context, Hierarchy and Scheme objects.

```
-create
```

```
-default_objects
```

```
=====
```

Description:

Creates a Classification Hierarchy node object.

```
-create
  -node
```

=====

Description:

Creates a new instance of Classification object.

```
-create
  -classification_object
```

=====

Description:

Creates a set of Classification objects.

```
-create
  -classification_objects
```

=====

Description:

Creates a Classification node hierarchy from Classification hierarchy.

```
-create
  -node_hierarchy
```

=====

ENVIRONMENT

This utility must be run in the Teamcenter shell environment.

FILES

As specified in *Log files produced by Teamcenter* and the **filetypeDefaults.txt** configuration file.

The entries in the **filetypeDefaults.txt** configuration file map the extensions of all imported files to specific Teamcenter data structures: GRM relationship type, dataset type, named reference, optional default tool, and optional subdirectory name. The mapping of file extensions to specific Teamcenter data structures can be configured per import directory/subdirectory combination.

smlutility

Updates shared Classification hierarchy definitions to all sites with which they are shared.

Use this command if you do not want to run the **subscriptionmgrd** daemon in the background. This utility can also be used to share specific definitions immediately, for example, if you modify a definition and want to share it with a colleague at a different site. In such cases, you can execute the command by specifying the definitions you want to share and the sites to which you want to update these definitions.

SYNTAX

```
smlutility [-u=user-id {-p=password | -pf=password-file} -g=group]

{[-delete | -list_hierarchy | -sync | -reassign_to_rev | -ask_shared_sites |
-add_shared_sites | -update_shared_sites | -list_local_icos | -migrate |
-update_unit_system | -lost_icos | -list_invalid_class_ids | -repair_default_value |
-get_history | -assign_pft | -assign_tp | -add_dict_attr | -mod_dict_attr | -mod_class |
-add_class_attr | -add_class_image | -add_view_attr | -remove_view_attr |
-create_view | -remove_views | -create_class | -set_ico_values -h]}
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-delete

Deletes definitions based on class, view, ICO ID, or attribute. For example:

```
smlutility -delete SA user SA password SA group
-class -id=class-ID [-icos] [-recurse] |
-view -id=class-ID [-icos] |
-ico (-cid=class-ID | -id=object-ID) |
-attribute -id=ID-number
-keylov
```

-class

Removes the identified class.

-icos

Removes all instances of the specified class.

-recurse

Removes all children of the specified class.

-view

Removes the specified view.

-icos

Removes all instances of the specified view if the view is a subclass.

-ico

Removes the identified instances (ICOs).

-attribute

Removes the specified attribute.

-keylov

Removes the specified key-LOV.

-list_hierarchy

Displays the classification hierarchy starting from the specified class.

For example:

```
smlutility -list_hierarchy [-u=SA user -p=SA password -g=SA group]
                        -id=class|groupID [-icos [-wso]] [-views]
                        [-attr] [-norecurse] [-nodescribe]
```

-icos

Displays basic information about the ICOs of each class.

-wso

Displays information about the classified workspace object.

-norecurse

Lists direct children only.

-nodescribe

Displays less information.

-sync

Synchronizes Classification definition data with remote sites, for example:

```
smlutility -sync SA-user SA-password SA-group [-definition] [-sites] [-
verbose]
```

-definition=

VIEW | CLASS | GROUP | DICTIONARY):ID [,ID[,...]]

VIEW:

Class-ID::View-ID

DICTIONARY:

Attribute-ID

CLASS:

Class-ID

KEYLOV:

KeyLOV-ID

GROUP:

Group ID

Note:

You can specify multiple definitions. If no definition is specified, all objects are synchronized.

Examples

`-definition=VIEW:myClass::myView,myClass::mySecondView`

`-definition=CLASS:myClass,mySecondClass,MyThirdClass`

`-definition=KEYLOV:-20000`

-sites

Specifies the sites to which the data is updated. If you do not specify sites, synchronization takes place between all sites with which definitions are shared. For example:

`-sites=site-name[, site-name[, ...]]`

Example

`-sites=IMC-12345,IMC-56789`

-reassign_to_rev

Modifies ICOs that classify an item so that they classify the latest item revision. For example:

```
smlutility -reassign_to_rev -u=SA-user -p=SA-password -g=SA-group class
-id=class ID [-recurse]|view -id=view-ID|ico -id=object-ID
```

class

Identifies the class for which all ICOs will be reassigned. For example, using **class -id=ugc010101** indicates that all ICOs in the **ugc010101** class that classify an item will be changed so that they classify the latest revision of the item. You can use the **-recurse** option to reassign all ICOs of the class and all subclasses of the identified class. For example, using **class -id=ugc010101 -recurse** indicates that all ICOs in the **ugc010101** class that classify an item will be changed so that they classify the latest revision of the item. In addition to changing the ICOs of the class, the ICOs in all subclasses of the given class are changed as well.

view

Reassigns all ICOs of the identified view. For example, using **class -id=ugc010101 view -id=DefaultView** reassigns all ICOs in the **DefaultView** of class **ugc010101** so that the latest revision of the item is classified.

ico

Reassigns the identified Classification instances (ICOs). For example, using **ico** **-id=ugc010101_001** changes the specific ICO (**ugc010101_001**) so that it classifies the latest revision of the item.

General rules

If an ICO to be reassigned has no associated item or already classifies an item revision, the ICO is not reassigned.

When an ICO is reassigned, the ICO object ID is changed automatically. For example, ICO **ugc010101_001** changes to **ugc010101_001/A** if **A** is the latest revision of the item.

-ask_shared_sites

Returns the names of the sites to which the specified object is shared.

```
smlutility -ask_shared_sites [-u=SA-user] [-p=SA-password] [-g=SA-group]
-objectType={class | view | attribute | key-LOV} -objectID=object-ID
```

-objectType

Specifies the type of object whose site you want to know. You can request the site of a class, view, attribute or key-LOV.

-objectID

Specifies the ID of the object whose site you request. If you request a view's site, the ID has the following format:

class-ID : :view-ID

-add_shared_sites

Sets the **Shared Site** property for a class, and optionally including its descendants and parents.

```
smlutility -add_shared_sites -u=user -p=password -g=group
-classes=class-ID -sites=site-name
-options=options
```

-classes

classid[,classid[,...]]

-sites

sitename[sitename[,...]]

-options

option[,option[,...]]

shareChildClasses

Specifies to share child classes.

shareDefaultViews

Specifies to share default views.

shareSubclasses

Specifies to share subclasses.

shareSpecificViews

Specifies to share specific views. The **ICS_share_viewtypes** preference, which can be set to a combination of **user**, **group**, or **role**, is evaluated.

shareParents

Specifies that the parents of the classes named in the argument **classes** are also shared. If this option is not set, the share operation fails for classes whose parents are not shared to the selected site.

shareViews

Use to share default views, subclasses, and specific views, that is, specifying **shareViews** includes the **shareDefaultViews**, **shareSubclasses**, and **shareSpecificViews** options.

shareAll

Use to include the **shareViews**, **shareChildClasses**, and **shareParents** options.

For example:

```
smlutility -add_shared_sites -classes=myClass1,myClass2 -sites=Vienna
           -options=shareChildClasses,shareDefaultViews
```

-update_shared_sites

Changes the site name in all groups, classes, views, attributes, and key-LOVs when updating a site name in Multi-Site. After running **smlutility** with the **update_shared_sites** option, you must remove the replica object from the old or obsolete site (if it still exists) and not perform any other operations on this object. Additionally, you must delete the **ImanExportRecord** for the deleted replica from the owning object.

```
smlutility -update_shared_sites [-u=SA-user] [-p=SA-password] [-g=SA-
group] -oldSite=site-name -newSite=site-name
```

-oldSite

Specifies the original site name that is to be replaced by the new name specified in **newSite** in all groups, classes, views, attributes and key-LOVs.

-newSite

Specifies the new site name to replace the old one specified in **oldSite** in all groups, classes, views, attributes, and key-LOVs.

If an empty value is specified, then the **oldSite** site is removed. If you pass an empty string as a value to **newSite** argument, the sharing is removed on the class definition.

-list_local_icos

Lists all ICOs that are owned locally.

```
smlutility -list_local_icos [-u=SA-user] [-p=SA-password] [-g=SA-group]
-cid=Class-ID [-verbose]
```

-cid

Specifies the unique ID of a classification class or group whose local ICOs should be listed.

-verbose

Displays additional information.

-migrate

Assigns the measurement system to each individual ICO within a class.

If an ICO previously exists within a solely metric or nonmetric class, the measurement system is not directly assigned to the ICO but is contained in the class definition. When you move from a metric or nonmetric class to one that contains both, Teamcenter assigns the measurement system to each individual ICO within the class.

```
smlutility -migrate [-u=SA-user] [-p=SA-password| -pf=password-file] [-g=SA
group] -cid=class-ID
[-dryrun] [-verbose={0|1|2}][ -remote]
```

-dryrun

Displays results of running the utility without making any changes and tests if all ICOs of the class can be accessed and changed.

-verbose

Displays additional information. The value for this argument must be one of the following:

- 0** No output
- 1** Output on error
- 2** Information and error output

-remote

Updates remote ICOs as well.

-update_unit_system

Changes a class or a class hierarchy that is previously metric or nonmetric to support both unit systems simultaneously.

```
smlutility -update_unit_system [-u=SA-user] [-p=SA-password] [-g=SA-group] -cid=class-ID -unit=both [-recurse]
```

-cid

Specifies the class ID of the class required to support both unit systems.

-unit=both

Specifies that the class is changed to support both unit systems.

-recurse

Changes all child classes beneath the class specified by the **-cid** parameter to support both unit systems.

-lost_icos

Reports and recovers ICOs that do not have a class ID set. If, during creation, Classification crashes, an ICO can already be created, but the attribute specifying in which class it is to be classified not yet saved. This utility finds these unsaved ICOs and can, optionally, classify them in the specified class.

```
smlutility -lost_icos [-u=user-id {-p=password | -pf=password-file} -g=group]
[-cid=class-ID]
[-createClass]
[-recover]
[-reportfile=name-of-external-file-containing-results]
```

-cid

Specifies the ID of the class in which the recovered ICOs are to be classified. The ICOs are only classified in this class if **-recover** is also specified.

-createClass

Creates the specified class if it does not already exist.

-recover

Moves the found ICOs to the specified class. If this argument is not set, Teamcenter lists the ICOs in the command window or in a file, if you set the **-reportfile** argument.

-reportfile

Specifies the name of a file in which Teamcenter writes the names of all the ICOs found by the lost ICO search. If you do not use this argument, Teamcenter lists the results in the command window.

-list_invalid_class_ids

Lists all the group and class IDs in the database that contain invalid characters. By default, the following characters are invalid:

```
|%*:(){} \
```


You can specify a subset of these characters to be valid using the **ICS_allowed_chars_for_class_id** preference.

```
smlutility -list_invalid_class_ids [-u=user-id {-p=password |
-pf=password-file} -g=group]
[-reportfile=file-name]
```

-reportfile

Specifies the name of a file in which Teamcenter lists all the invalid group and class IDs. If you do not use this argument, Teamcenter lists the results in the command window.

-repair_default_value

Deletes default value for a view attribute if the corresponding class attribute's value is set to fixed.

-get_history

Reports what specific Manufacturing Resource Library (MRL) modules (including version information) are imported into the database.

```
smlutility -get_history -u=user-id {-p=password | -pf=password-file} -g=group
```

Example:

```
smlutility -get_history -u=my_user -p=my_password -g=dba
```

When you start this command, you get the following output:

```
20150311135828|MM|ENG|MRL|CATALOG|N|3.1.4|V10000.1.0.20130604
20150311135836|MM|ENG|MRL|TOOLS|N|3.1.4|V10000.1.0.20130604
20150311135841|MM|ENG|MRL|MACHINES|N|3.1.4|V10000.1.0.20130604
20150311135845|MM|ENG|MRL|FIXTURES|N|3.1.4|V10000.1.0.20130604
20150311135848|MM|ENG|MRL|CAM_TEMPLA|N|3.1.4|V10000.1.0.20130604
20150311135852|MM|ENG|MRL|WELDGUNS|N|3.1.4|V10000.1.0.20130604
20150311135857|MM|ENG|MRL|ROBOTS|N|3.1.4|V10000.1.0.20130604
20150311135902|MM|ENG|MRL|FACTORY RE|N|3.1.4|V10000.1.0.20130604
20150311135907|MM|ENG|MRL|FACTORY CO|N|3.1.4|V10000.1.0.20130604
20150311135912|MM|ENG|MRL|MACHINING|N|3.1.4|V10000.1.0.20130604
20150424090758|MM|ENG|MRL|TOOLS|N|3.1.4|V10000.1.0.20130604
20150424090800|MM|ENG|MRL|TOOLS|Y|3.1.4|V10000.1.0.20130604
20150424090805|MM|ENG|MRL|MACHINES|N|3.1.4|V10000.1.0.20130604
20150424090808|MM|ENG|MRL|MACHINES|Y|3.1.4|V10000.1.0.20130604
20150424090810|MM|ENG|MRL|CAM_TEMPLA|N|3.1.4|V10000.1.0.20130604
```

Note:

If no MRL was imported into the database, you get no output.

Column output definitions:

```
20150311135836|MM|ENG|MRL|TOOLS|N|3.1.4|V10000.1.0.20130604
```

20150311135836

Indicates the time when this module was imported. (The format is *yyyymmddhhmmss*.)

MM

Indicates the measurement unit of this module; it could be *MM* (metric) or *INCH* (nonmetric); MRL uses both classes, which can store metric and nonmetric classification objects, and writes **MM** in the history.

ENG

Indicates the locale of the module that was imported, for example, *ENG* (English).

MRL

Indicates the type of the resource library kit; possible values are *MTL*, *TCMFG_CONTENT_KIT*, and *MRL*.

TOOLS

Indicates the name of the module, for example, *TOOLS* = tools, *MACHINES* = machines, *FACTORY CO* = factory conveyors.

N

Indicates if sample data was imported for this module, values are **N** (no) and **Y** (yes).

3.1.4

Indicates the MRL version.

V10000.1.0.20130604

Indicates the Teamcenter version.

-assign_pft

Assigns a part family template file to the specified class. This utility provides the same features as assigning a part family template in the user interface.

```
smlutility -assign_pft -u=user-id {-p=password | -pf=password-file} -g=group
-classID=class-ID
-itemID=item-ID
-expressionList={expression1,expression2...}
[-attachRevision]
[-enabled=true|false]
[-useForChildClasses=true|false]
[-noApplicability]
[-default]
[-insert]
```

-classID

Specifies the classification class to which the part family template is to be attached.

-itemID

Specifies the ID of the item containing the NX part family template.

-expressionList

Specifies the list of NX expressions that are mapped to the classification class attributes.

-attachRevision

Specifies that the item revision of the part family template is attached to the class. If this argument is not set, the item of the part family template is attached by default.

-enabled

Specifies that the part family template is available for graphic generation. If this argument is not set, the part family template is attached but cannot be used to generate graphics.

-useForChildClasses

Makes the part family template available for child classes to use to generate graphics.

-noApplicability

Specifies that applicability is not set for the mapped attributes. If you do not set this argument, a default view must be available for the class. The **Graphics Creation** applicability is set for all the mapped expressions (the ones listed in the **-expressionList** argument) in the default view.

-default

If there are multiple part family templates associated with a class, sets the current template as the default template when generating graphics.

-insert

Retains all part family templates that were previously attached to the class after the utility is run. Not setting this argument removes all other attached part family templates from a class before attaching the new one.

-assign_tp

Assigns a template part to the specified class. This utility provides the same features as assigning a template part in the user interface.

```
smlutility -assign_tp -u=user-id {-p=password | -pf=password-file} -g=group
-classID=class-ID
-itemID=item-ID
-expressionList={expression1,expression2...}
[-attachRevision]
[-enabled=true|false]
[-useForChildClasses=true|false]
[-noApplicability]
[-default]
[-insert]
```

-classID

Specifies the classification class to which the template part is to be attached.

-itemID

Specifies the ID of the item containing the NX template part.

-expressionList

Specifies the list of NX expressions that are mapped to the classification class attributes.

-attachRevision

Specifies that the item revision of the template part is attached to the class. If this argument is not set, the item of the template part is attached by default.

-enabled

Specifies that the template part is available for graphic generation. If this argument is not set, the template part is attached but cannot be used to generate graphics.

-useForChildClasses

Makes the template part available for child classes to use to generate graphics.

-noApplicability

Specifies that applicability is not set for the mapped attributes. If you do not set this argument, a default view must be available for the class. The **Graphics Creation** applicability is set for all the mapped expressions (the ones listed in the **-expressionList** argument) in the default view.

-default

If there are multiple template parts associated with a class, sets the current template as the default template when generating graphics.

-insert

Retains all template parts that were previously attached to the class after the utility is run. Not setting this argument removes all other attached template parts from a class before attaching the new one.

-add_dict_attr

Creates dictionary attributes.

```
smlutility -add_dict_attr -u=user-id {-p=password | -pf=password-file}
-g=group
-attr_id=attribute-ID
-name=attribute-name
-format=attribute format
```

-attr_id

Specifies the ID of the new dictionary attribute.

-name

Specifies the name of the new dictionary attribute.

-format

Specifies the **format** (for example, string, boolean, or integer) of the new dictionary attribute.

-mod_dict_attr

Modifies the value of the given dictionary attribute. You can specify only one parameter at a time.

```
smlutility -mod_dict_attr -u=user-id {-p=password | -pf=password-file}
-g=group
-attr_id=attribute-ID
-property=[{NAME|SHORT_NAME|UNIT1|UNIT2|FLAGS|USER1|USER2|DESCRIPTION|
ANNOTATION|COMMENT|FORMAT1|FORMAT2|DEFAULT_VALUE|DEFAULT_VALUE2|
MIN_VALUE|MAX_VALUE|MIN_VALUE2|MAX_VALUE2|DEPENDENCY_CONFIGURATION}]
-value=value-to-set
```

-attr_id

Specifies the ID of the modified dictionary attribute.

-property

Specifies the name of the dictionary attribute property to be modified. The name of the property must be entered as listed.

-value

Specifies the new value of the dictionary attribute.

You must run this command for each property to be modified.

-mod_class

Modifies class properties. In general, these are the attributes found on the **Class Details** tab.

```
smlutility -mod_class -u=user-id {-p=password | -pf=password-file} -g=group
-cid=class-ID
[-name=class-name]
[-short_name=class-short-name]
[-alias_name=alias name [-alias_ref=alias-reference]]
[-alias_lang=alias-language]
[-unit={both|metric|non-metric}]
[-abstract={true|false}]
[-assembly={true|false}]
[-user1=user-1-text]
[-user2=user-2-text]
[-dependency=dependency-configuration-text]
[-tclScriptEvaluation={true|false}]
[-verbose]
[-recurse]
```

-add_class_attr

Adds or modifies attributes in a classification class. In general, these are the attributes that are displayed on the **Class Attributes** tab. If you use this argument for existing attributes, their values are overwritten with the values specified.

```
smlutility -add_class_attr -u=user-id {-p=password | -pf=password-file}
-g=group
[-array=array-size]
[-isArray={true|false}]
[-mandatory={true|false}]
[-unique={true|false}]
[-propagated={true|false}]
[-user_def_button={true|false}]
[-protected={true|false}]
[-annotation=attribute-annotation]
[-defaultMetric=metric-value]
[-defaultNonmetric=nonmetric-value]
[-applicability1={true|false}]
[-applicability2={true|false}]
[-applicability3={true|false}]
[-applicability4={true|false}]
[-applicability5={true|false}]
[-user1=user-1-text]
[-user2=user-2-text]
[-dep_attr=dependency-attribute]
[-dep_config=dependency-configuration-useKeyLOVLevel(1),
useKeyLOVLevel(2)...]
[-localValue={true|false}]
[-minMetric=metric-minimum-value]
[-maxMetric=metric-maximum-value]
[-minNonmetric=nonmetric-minimum-value]
[-maxNonmetric=nonmetric-maximum-value]
[-moveup]
[-verbose]
```

-add_class_image

Adds or updates class images.

```
smlutility -add_class_image -u=user-id {-p=password | -pf=password-file}
-g=group
-cid=class-ID
[-classImagePath=path-to-class-image]
[-classImageRef={ICS-ClassImage1|ICS-ClassImage2|...ICS-ClassImage8}]
[-classIconPath=path-to-class-icon]
[-verbose]
```

-cid

Specifies the unique ID of a classification class to which the image should be added.

-classImagePath

Updates the class image path. If **classImageRef** is not set, the image is set as first image. If a class image is already defined, it is replaced with the new one specified here.

-classImageRef

Sets the order in which the image is displayed. Possible options are **ICS-ClassImage1...ICS-ClassImage8**. If there is already an image at the specified position, it is replaced with the new image.

-classIconPath

Updates the class icon path. If a class icon is already defined, it is replaced with the new one specified here.

-add_view_attr

Adds or modifies attributes in a view. These are generally the attributes found in the **Add View** tab.

- If the **-view_Type** argument is not set, the attributes in all the following default view types are added or modified:
 - **Default**
 - **User**
 - **Group**
 - **Role**
 - **Project**

If the attribute already exists in the specified view, its value is updated.

```
smlutility -add_view_attr -u=user-id {-p=password | -pf=password-file}
-g=group
-cid=class-ID|view-ID
-attr_id=attribute-ID
[-array=array-size]
[-mandatory={true|false}]
[-unique={true|false}]
[-protected={true|false}]
[-keep_width={true|false}]
[-user1=user-1-text]
[-user2=user-2-text]
[-defaultMetric=metric-value]
[-defaultNonmetric=nonmetric-value]
[-minMetric=minimum-metric-value]
[-maxMetric=maximum-metric-value]
[-minNonmetric=minimum-nonmetric-value]
[-maxNonmetric=maximum-nonmetric-value]
```

```

[-applicability1={true|false}]
[-applicability2={true|false}]
[-applicability3={true|false}]
[-applicability4={true|false}]
[-applicability5={true|false}]
[-dep_attr=dependency-attribute]
[-dep_config=dependency-configuration-useKeyLOVLevel(1),
  useKeyLOVLevel(2)...]
[-view_type={Default|User|Group|Role|Project|Mapping|NXLibCriteria|
NXLibResult|NXLibRetrieve}]
[-position={first|last|after|before}]
[-attr_list=attribute ids]
[-adjust_layout_tags]
[-text_pos={pre|post}]
[-verbose]
[-recurse]
[-reportfile=path-to-output-file-(default:stdout)]

```

-remove_view_attr

Removes attributes from a view.

If the **-view_Type** argument is not set, the specified attribute in all the following default view types are removed:

- **Default**
- **User**
- **Group**
- **Role**
- **Project**

```

smlutility -remove_view_attr -u=user-id {-p=password | -pf=password-file}
-g=group
-cid={class-ID|view-ID}
-attr_id=attribute-ID
[-view_type={Default|User|Group|Role|Project|Mapping|NXLibCriteria|
NXLibResult|NXLibRetrieve}]
[-verbose]
[-recurse]
[-reportfile=path-to-output-file-(default:stdout)]

```

-create_view

Creates a new view and associates it with a specified class, group, or role.

```

smlutility -create_view -u=user-id {-p=password | -pf=password-file}
-g=group

```



```
-view_name= name-of-view
-view_type={default|user|group|role}
-view_id= {User-view-ID (must exactly match Teamcenter user ID)|Role-view-ID (must exactly match Teamcenter role ID)|Group-view-ID (must exactly match Teamcenter group ID)}
[-attr_ids= attribute-IDs-to-add-to-view]
```

The attribute IDs must be in a comma-separated list with no spaces.

-remove_views

Removes views associated with a specified class or group.

```
smlutility -remove_views -u=user-id {-p=password | -pf=password-file}
[-g=group]
[-view_type=Default|User|Group|Role|Project|Mapping|NXLibCriteria|
NXLibResult|NXLibRetrieve][-verbose]
[-recurse]
[-reportfile=path-to-output-file-(default:stdout)]
```

If **-view_Type** is not set, the default view types are removed: **Default**, **User**, **Group**, **Role**, and **Project**.

Examples:

To delete the **ABC** view from the **TC_Test** class:

```
smlutility -remove_views -class_id=TC_Test -viewid="ABC"
```

To delete all group views from the **TC_Test** class:

```
smlutility -remove_views -cid=TC_Test -view_type=Group
```

To delete all group views from the **TC_Test** class, including all child classes:

```
smlutility -remove_views -cid=TC_Test -view_type=Group -recurse
```

To delete all group views that begin with the letters **ABC** from the **TC_Test** class, including child classes:

```
smlutility -remove_views -cid=TC_Test -viewid="ABC*" -view_type=Group -recurse
```

-set_ico_values

```
smlutility -set_ico_values -u=user-id {-p=password | -pf=password-file} -g=group
-icoId=ICO ID -attrIds=comma separated list of attribute IDs
-attrValues=comma separated list of attribute values
```

Only ICOs with unique IDs can be updated. Variable length array (VLA) properties are not supported.

-create_class

```
smlutility -create_class -u=user-id {-p=password | -pf=password-file} [-g=group]
-cid=class-ID -class_name=name-of-new-class -class_type={group |
abstract_class | storage_class}
[-unit_system={metric | nonmetric | both (both is default)}]
[-attr_ids=attribute-IDs-to-add-to-class]
```

The attribute IDs must be in a comma-separated list with no spaces.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility synchronizes shared definitions but does not change to which sites definitions are shared. If you specify a definition and site for synchronization, but the site is not one for which sharing has been designated, the definition is not sent to the site. For example, if the **myClass** class is only shared to the sites **Rome** and **London**, nothing is synchronized if you execute the following command:

```
smlutility -sync SA user SA password SA group -definition=CLASS:myClass
-sites=Paris
```

EXAMPLES

- To import Classification data from a file, perform the following steps:

1. Enter the following command at a system command prompt:

```
smlutility -import user password group file -insert
```

For example:

```
smlutility -import my_user my_password dba
in-CLASSexample.sml -insert
```

2. Press the Enter key.

- To update existing Classification data, perform the following steps:

1. Enter the following command at a system command prompt:

```
smlutility -import user password group file -update
```

For example:

```
smlutility -import my_user my_password dba
in-CLASSexample.sml -update
```

2. Press the Enter key.

- Perform the following steps to remove a class definition:

1. Enter the following command at a system command prompt:

```
smlutility -delete user password group [class|view|instance] [id]
```

For example:

```
smlutility -delete my_user my_password dba class screws
```

2. Press the Enter key.

- Perform the following steps to remove a specific attribute with ID 1643:

1. Enter the following command at a system command prompt:

```
smlutility -delete user password group attribute -id=ID number
```

For example:

```
smlutility -delete my_user my_password dba attribute -id=1643
```

2. Press the Enter key.

- Perform the following steps to remove all unreferenced attributes:

1. Enter the following command at a system command prompt:

```
smlutility -delete my_user my_password dba attribute -id=*
```

2. Press the Enter key.

About smlutility and sml file format

Overview of smlutility and SML file format

The **smlutility** program provides a variety of utilities related to Classification, including the capability to import and export the Classification hierarchy and related instances to and from an ASCII text format. These import/export features are useful for the bulk loading of Classification data and for transferring a Classification hierarchy tree from one database to another.

Note:

The **-import** and **-export** arguments of the **smlutility** utility are deprecated and do not support newer features. To ensure that your data is imported and exported accurately, use PLM XML.

Although import/export data can be contained in one file or distributed over multiple files, the order in which the elements are imported is important. For example, attribute definitions can reference lists. Therefore, the lists must already be defined. Additionally, classes and subclasses make reference to attributes that must first be defined in the attribute (UNCT) dictionary. To avoid dependency problems, Siemens Digital Industries Software recommends that the following elements be imported in the following order:

- List
- Attribute
- Class
- Subclass
- Data

Each line of the import/export file starts with a three-character keyword followed by the appropriate arguments. Each field of the command line is separated by the vertical bar (|) character.

For example:

```
Keyword | argument1| argument2| argument3|
```

The following table lists reserved characters, their purpose, and a description:

Character	Purpose	Description
! (exclamation mark)	Comment	All characters after the ! character are comments. You can place the ! symbol anywhere within the command line.
& (ampersand)	Line continuation	Continues the command on the following line. Place the & symbol at the end of a line.
(vertical bar)	Field separation	Separator for the arguments.

Elements of an import/export file

An import/export file contains the following elements:

File header	A file header is not required for import files, however, it is recommended. The file header includes information that helps you identify the data contained within the file. The file header is information-only. The export function automatically generates a file header. The export file header contains information, such as the Teamcenter version number, user, node, database server, and creation date.
List definitions	A list of values is associated with an attribute, such that when a user clicks the down-arrow in the Attribute field, the system displays a list of valid values for that attribute. Teamcenter uses lists when an attribute has a finite set of legal values. The list is more efficient than manual entry because it eliminates the need for the user to memorize the valid values. Lists also enforce consistency. Lists are defined prior to the attribute definition and are associated with the attribute at the time of its definition. A list definition is composed of the menu definition (STV keyword) and one or more menu item definitions (STD keyword).
Attribute definitions	Attributes are placeholders for values that distinguish one instance of a class/subclass from another. For example, within the sheet metal screws subclass, the length, diameter, and thread attributes distinguish one screw from another. Attributes are defined prior to the class definition and are associated with the class at the time of its definition. Once defined, attributes, like lists, are stored in a dictionary. They are only defined once and can be used as many times as needed. Attribute definitions use the SMV keyword.
Group definitions	Groups are defined in the import/export file to organize a large set of classes. Groups are essentially classes with no attributes.

Group definitions use the **SML** keyword.

Class definitions

The class section of the import/export file defines the class, associates the class with a group, and associates a list of attributes with the class.

A class definition is composed of the class (**SML** keyword) and the association of one or more attributes using the **STD** keyword.

Subclass definitions

The subclass section of the file defines subclasses, associates each subclass with a class, and associates a subset of the class attributes with each subclass.

A subclass definition is composed of the subclass (**BLD** keyword) and the association of one or more attributes using the **BSM** keyword.

Instances

Instances of Classification objects are normally created by classifying a Teamcenter object, but can also be imported or exported. This is useful when loading third-party part or tool libraries, copying a Classification scheme from one database to another, or performing bulk loads of existing data.

Instances of Classification objects are created using the **DAT** keyword.

SML import file and BOM line syntax

The following figure illustrates the SML file format required to support import of assembly structures.

Assembly structure BOM line syntax example:

```
DAT |mc0101_001|mc01|01|0| | |2005: 01
BOM |mc0101_001|mc01|01| &
  N0;          1;mc0101_002;mc01;01; 1;1; 0.0;0.0;0.0;90.0;-0.0;-0.0;| &
  N0.0;        2;mc0101_003;mc01;01; 1;1; -54.0;-84.0;0.0;-0.0;-0.0;90.0;| &
  N0.0.0;      6;mc0101_004;mc01;01; 1;1;
-47.227;-77.227;0.0;-0.0;-0.0;-175.0;| &
  N0.0.0.0;    7;C1;;; 1;1; +0.0;+0.0;+0.0;+0.0;+0.0;+0.0;| &
  N0.0.1;      2;mc0101_005;mc01;01; 1;1; 10.0;-104.0;0.0;-0.0;-0.0;90.0;| &
  N0.0.1.0;    6;mc0101_006;mc01;01; 1;1;
3.121;-97.121;1.590;-0.0;-0.0;95.0;| &
  N0.0.1.0.0;  7;C2;;; 1;1; +0.0;+0.0;+0.0;+0.0;+0.0;+0.0;|
```

As shown in this example, each line containing an assembly structure definition begins with a BOM tag. The three columns that follow the BOM tag contain the record, class, and subclass identifiers of the database record that defines a resource assembly. The remaining columns contain information pertaining to different components of the assembly.

The record identifier is equivalent to the ICO-ID and the item ID.

The first entry encodes the position of the component within the hierarchical assembly structure.

The second entry defines the node type for the component. Supported node types are:

Node type value	Node type
1	Root Component
2	Intermediate Component
6	Bottom level Component
7	Propagation Start Point

For component nodes, the third, fourth and fifth entries contain the record, class, and subclass identifiers of the database record representing the component. For propagation start points (PSPs), the third entry contains the number of the propagation start point prefixed by the letter C. In this case, the fourth and fifth entries remain empty.

The sixth entry contains a component quantity field. Multiple components of the same type, for example **mc0101_002**, must be listed on subsequent lines, each indicating a quantity of one, which allows each line to define its own component transformation.

The seventh entry indicates whether the component graphics are to be displayed in the context of the Genius4000 XTM assembly. This value is not yet mapped to any attribute of the ICS data model.

The remaining entries, including and beyond the eighth entry, define the position of a component in terms of coordinates **x**, **y** and **z** as well as the orientation of the component using the Euler rotation angles **a**, **b** and **c**. This information generally applies only to a single component, thus diluting the interpretation of the component quantity field.

Import/export file example

```
!#####
!  
! Classification Example  
!  
! This file includes the definition of  
! - attributes  
! - menus  
! - groups  
! - classes/subclasses  
! - and some sample data records  
!  
!#####  
!  
! ICM -> ICM Classification Root  
! myhl -> My Highest Level  
! myg1 --> Subgroup 1
```

```

! 2001 2002 2003 2004 2005 2006 2007
! mc01 --> My Class 1 Diam Thk Descr Angl Dir
! mc01:00 --> All Data X X X
! mc01:01 --> Subclass 1-01 X X X X
! mc01:02 --> Subclass 1-02 X X X X
!
! mc02 --> My Class 2 Diam Thk #Hls Descr Mat AnglDir
! mc02:00 --> All Data X X X
! mc02:01 --> C2 with material X X X X X
! mc02:02 --> C2 including all X X X X X X X
! mc02:03 --> C2 limited X X X
!
! myg2 --> Subgroup 2
! ...
!
!#####
! ---- Attributes ----
!SMV| UNCT| Format F.2| Text-Description| Short-Text|Ref|Unit U.2|Fl|EX1|
EX2|
SMV | 2001| 20307 20509| Diameter | Diam | |mm inch| 0| | |
SMV | 2002| 20408 20509| Thickness | Thickn. | |mm inch| 0| | |
SMV | 2003| 10002 0| Number Of Holes | Num Holes | | | 0| | |
SMV | 2004| 00080 0| Description | Descr. | | | 0| | |
SMV | 2005| -2005 0| Material | Mat | | | 0| | |
SMV | 2006| 20307 0| Angle | Angle | |degree | 0| | |
SMV | 2007| -2007 0| Direction | Dir | | | 0| | |
! ---- Popups ----
! Popup for Material
STV |-2005| Material Popup |
STD | 01 | Lexan |
STD | 02 | Alloy Steel |
STD | 03 | HS Steel |
STD | 04 | Aluminium |
STD | 05 | Copper |
! Popup for Direction
STV |-2007| Direction Popup |
STD | N | Neutral |
STD | L | Left |
STD | R | Right |
! ---- Groups ----
!SML| SMLID | Typ | Group |Description |ShortDesc|Graphic|Flags|EX1|EX2|
SML | myh1 | SAM | ICM |My highest level | | | 2 | | |
SML | myg1 | SAM | myh1 |Subgroup 1 | | | 2 | | |
SML | myg2 | SAM | myh1 |Subgroup 2 | | | 2 | | |
! --- Classes ---
! Class mc01 : My Class 1 (UNIT: english) -----
!SML| SMLID | Typ | Group |Description |ShortDesc|Graphic|Flags|EX1|EX2|
SML | mc01 | ICM | myg1 |My Class 1 | | | 1 | | |
! Attributes for "My Class 1"

```



```

!SMD| UNCT | ID | min | max | Flags | EX1 | EX2 |
SMD | 2001 | D1 | | | 0 | | |
SMD | 2002 | TH | | | 0 | | |
SMD | 2004 | REM| | | 0 | | |
SMD | 2006 | A1 | | | 0 | | |
SMD | 2007 | DIR| | | 0 | | |
! Subclass 00 : All Data for "My Class 1"
!
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 00 | All Data | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2001 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! Subclass 01 : "My Class 1" Subclasses 01
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 01 | My Subclass 1-01 | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2001 | | | 0 | | 0 | | 0 | | | |
BSM | 2006 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! Subclass 02 : "My Class 1" Subclasses 02
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 02 | My Subclass 1-02 | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2001 | | | 0 | | 0 | | 0 | | | |
BSM | 2002 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! Class mc02 : My Class 2 (UNIT: metric) -----
!SML| SMLID | Typ | Group |Description |ShortDesc|Graphic|Flags|EX1|EX2|
SML | mc02 | ICM | myg1 |My Class 2 | | | 0 | | |
! Attributes for "My Class 2"
!SMD| UNCT | ID | min | max | Flags | EX1 | EX2 |
SMD | 2001 | D1 | | | 0 | | |
SMD | 2002 | TH | | | 0 | | |
SMD | 2003 | NH | | | 0 | | |
SMD | 2004 | REM| | | 0 | | |
SMD | 2005 | MAT| | | 0 | | |
SMD | 2006 | A1 | | | 0 | | |
SMD | 2007 | D2 | | | 0 | | |
! Subclass 00 : All Data for "My Class 2"
!
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 00 | All Data | | | 0 | | |

```

```

!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2003 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! Subclass 01 : "My Class 2" C2 with material
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 01 | C2 with material | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2001 | | | 0 | | 0 | | 0 | | | |
BSM | 2003 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2005 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! Subclass 02 : "My Class 2" C2 including all
! (Attributes displayed in different order)
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 02 | C2 including all | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2005 | | | 0 | | 0 | | 0 | | | |
BSM | 2001 | | | 0 | | 0 | | 0 | | | |
BSM | 2002 | | | 0 | | 0 | | 0 | | | |
BSM | 2006 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
BSM | 2003 | | | 0 | | 0 | | 0 | | | |
! Subclass 03 : "My Class 2" C2 limited
!BLD| BLDID| Description |ShortDesc|Graphic|Flags|EX1|EX2|
BLD | 03 | C2 limited | | | 0 | | |
!
!BSM|SML-Id| min| max| Flag| TXT | Flag1| TXT1| Flag2| TXT2| EX1| EX2|
BSM | 2003 | | | 0 | | 0 | | 0 | | | |
BSM | 2004 | | | 0 | | 0 | | 0 | | | |
BSM | 2007 | | | 0 | | 0 | | 0 | | | |
! ---- Data Records ----
! Class 1
DAT |mc0101_001|mc01|01|0| | |2001: 3.75 |2004:My sample record |2007:L
DAT |mc0101_002|mc01|01|0| | |2001:15.5 |2004:Please remind me
DAT |mc0101_003|mc01|01|0| | |2001:85 |2004:Right oriented |2007:R
DAT |mc0101_004|mc01|01|0| | |2001: .123|2006: 90.0 |2007:L
DAT |mc0101_005|mc01|01|0| | | 2006:180 |2007:N
DAT |mc0101_006|mc01|01|0| | |2001:98.76 |2006:270 |2007:L
DAT |mc0102_001|mc01|02|0| | |2001: .01 |2004:Small data |2002:2.5
DAT |mc0102_001|mc01|02|0| | |2001: 0.333|2004:Negative Diameter|2002:4
DAT |mc0102_001|mc01|02|0| | |2001:20 | 2002:8.8
DAT |mc0102_001|mc01|02|0| | |2007:L |2004:Big Part |2002:13.73
! Class 2

```

```

DAT |lexan_01 |mc02|01|0| | |2001: 4.0 |2003: 4 |2005:01 |2007:L
DAT |lexan_02 |mc02|01|0| | |2001: 4.0 |2003: 6 |2005:01 |2007:L
DAT |lexan_03 |mc02|01|0| | |2001: 4.0 | 2005:01 |2007:R
DAT |lexan_04 |mc02|01|0| | |2001: 4.0 |2003: 4 |2005:01 |2007:R
DAT |lexan_05 |mc02|01|0| | |2001: 4.0 |2003:18 |2005:01 |2007:L
DAT |copper_01 |mc02|01|0| | |2001: 4.0 |2003: 4 |2005:05 |2007:L
DAT |copper_02 |mc02|01|0| | |2001: 4.0 |2003: 6 |2005:05 |2007:L
DAT |copper_03 |mc02|01|0| | |2001: 2.0 | 2005:05 |2007:L
DAT |copper_04 |mc02|01|0| | |2001: 2.0 |2003: 4 |2005:05
DAT |copper_06 |mc02|01|0| | |2001: 1.0 |2003:18 |2005:05 |2007:N
DAT |alu_01 |mc02|01|0| | |2001:18.1 |2003: 7 |2005:04 |2007:R
DAT |alu_02 |mc02|01|0| | |2001:20.2 |2003: 6 |2005:04 |2007:L
DAT |alu_03 |mc02|01|0| | |2001:21.3 | 2005:04 |2007:N
DAT |mc0202_001|mc02|02|0| | |2001:123.456 &
|2002:78.905 &
|2003: 13 &
|2005: 02 &
|2006: 45 &
|2007: N &
|2004: Keep in stock !
DAT |mc0203_001|mc02|03|0| | |2003: 2 |2004:Example 001 |2007:L
DAT |mc0203_002|mc02|03|0| | |2003: 4 |2004:Example 002 |2007:R
DAT |mc0203_003|mc02|03|0| | |2003: 8 |2004:Example 003 |2007:R
DAT |mc0203_004|mc02|03|0| | |2003: 16 |2004:Example 004 |2007:N
DAT |mc0203_005|mc02|03|0| | |2003: 31 |2004:Example 005 |2007:N
DAT |mc0203_006|mc02|03|0| | |2003: 62 |2004:Example 006 |2007:N
DAT |mc0203_007|mc02|03|0| | |2003: 88 |2004:
DAT |mc0203_008|mc02|03|0| | |2003: 56 |2004:Example 008 |2007:L
DAT |mc0203_009|mc02|03|0| | |2003: 39 |2004:Example 009 |2007:N
DAT |mc0203_010|mc02|03|0| | |2003: 27 |2004:Example 010 |2007:L

```

smlutility keyword syntax and description

STV and STD keywords (lists)

The **STV** keyword defines a list. Lists associated with an attribute provide a method by which the Classification administrator defines a list of values for a given attribute.

One or more **STD** statements defining the list of values for the list follow the **STV** statement defining the list.

Element name	Definition	Description
STV	Keyword	
STXT-ID	List ID number	Always negative.
Name	List title	

Element name	Definition	Description
STD	Keyword	Key for list.
Key		
Value	Text for list item	

The following example creates a list to be used with a **Cut Direction** attribute. The resulting list contains the **Left Hand Climb** and **Right Hand Climb** options.

Keyword	STXT-ID	Name
STV	-9110	Cut Direction

Keyword	Key	Value
STD	LHC	Left Hand Climb
STD	RHC	Right Hand Climb

SMV keyword (attributes)

The **SMV** keyword defines an attribute and adds the attribute definition to the attribute (UNCT) dictionary. Once an attribute is defined in the dictionary, it can be used and reused as needed.

Element name	Definition	Description
SMV	Keyword	Uniquely identifies an attribute and is the key field for associating the attribute with a class or subclass. UNCT numbers can be positive or negative. Note: Positive numbers from 0 to 999 are reserved for Siemens Digital Industries Software.
UNCT	Attribute identifier	
Format	Value format	

Element name	Definition	Description
		Time List
		Zero, one, or two formats can be entered in the Format field. .
Name	Attribute name	Describes the attribute that is being defined. The name can be a maximum of 63 alphanumeric characters in length and is case sensitive.
Short Name	Short name	Short name for an attribute can consist of up to 10 alphanumeric characters. When there is limited space available, Teamcenter uses the short name for creating reports.
Ref		Not used.
Units	Units	Units are entered as a string that appear after the attribute value in the Classification pane located on the Properties tab. Zero, one or two unit descriptions can be entered in the Units field.
Flags		Not used, always 0.
EX1		Not used.
EX2		Not used.

Units and formats

To identify the units and control the format of a numeric attribute, use the **Units** and **Format** parameters of the **SMV** command line.

For numeric attributes that change between measuring systems dependent on the class to which they are assigned, you can enter two formats and two units.

For example, if the numeric attribute **diameter** is used in the **inch** class and **metric** class, the following definition can be used:

```
SMV | -2491 | 21309 21308 | Diameter | Dia | | mm inch | 0 | |
```

This definition describes a **+/-6.3** format with units of **mm** for the **metric** class and a **+/-3.3** format with units of **inch** for the **inch** class. For example:

For + or - REAL(6.3) use 21311. Where 11 is the sum of: (6 plus 3=9) plus .=10 plus - which gives 11.

For a forced pos. REAL(6.3) use 20310. Where (6 plus 3 plus .)=10.

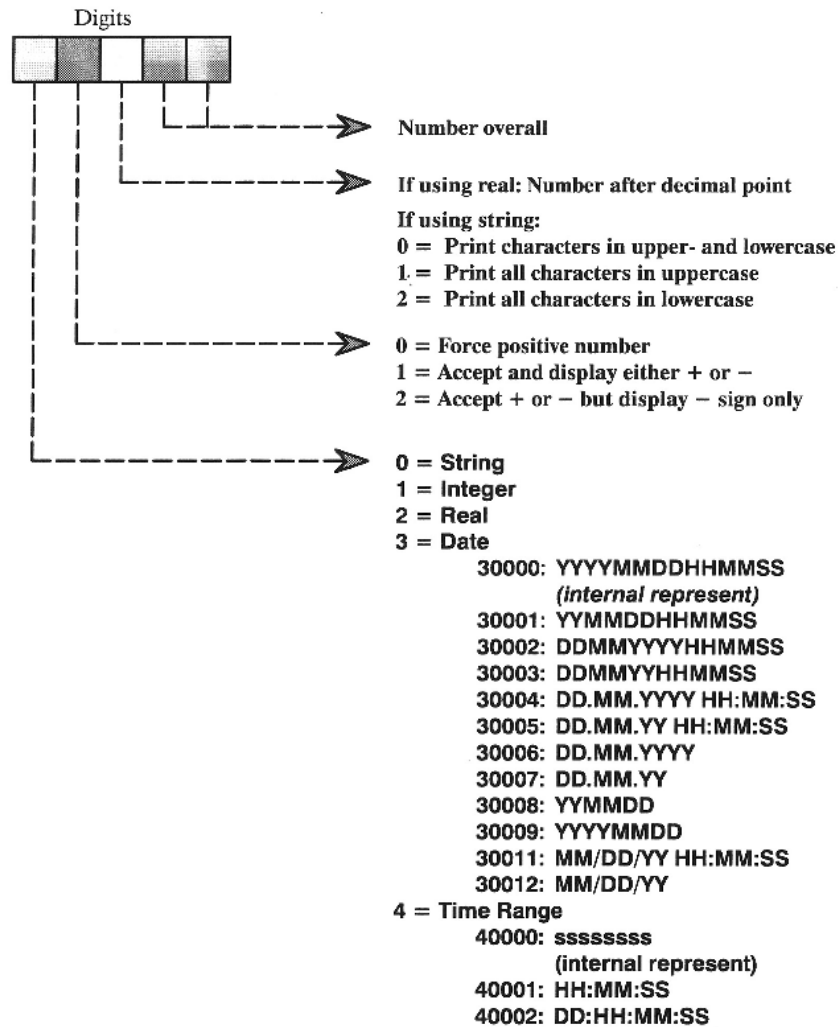
For a Real(3.3) that forces pos. numbers, use the code 20307.
 For a Real(6.3) that forces pos. numbers, use the code 20310.

The value in the **Flags** parameter of the **SML** (class) command determines which of the two definitions are used with that class.

Note:

Negative format numbers are the STXT-ID of a list.

The first digit of a positive format number defines what types of values are expected in the field and controls the format of the value. The format number can be up to five digits in length. The following figure explains the meaning of the digit in the format number:



Digit in format number

In the following tables, the system creates a list with two entries and then defines two **Cut Direction** attributes. The first **Cut Direction** attribute (UNCT number 3001) represents a string attribute that allows the user to enter a 40-character text string representing the cut direction. In this case, the user is responsible for entering valid strings for the attribute value within the Classification interface.

In the second attribute definition (UNCT number 3002) creates a **Cut Direction** attribute that allows the user to choose predefined values from a list.

Note:

The negative format number matches the STXT-ID of the list definition and associates the attribute to the previously defined list.

List definition

Keyword	STXT-ID	Name
STV	-3002	Direction
Keyword	Key	Value
STD	L	Left
STD	R	Right

Cut direction attribute definition

Keyword	UNCT	Format	Name	Short Name	Ref	Units	Flags
SMV	3001	00040 0	Cut Direction	Cut Dir			0
SMV	3002	-3002 0	Cut Direction	Cut Dir			0

SML and SMD keywords (class/group)

The **SML** keyword defines a class or group. A class definition consists of an **SML** definition statement followed by up to 60 **SMD** lines to associate attributes with the class. A group is essentially a class with no attributes; the group definition consists of an **SML** statement only with the **Flags** parameter set to **2**.

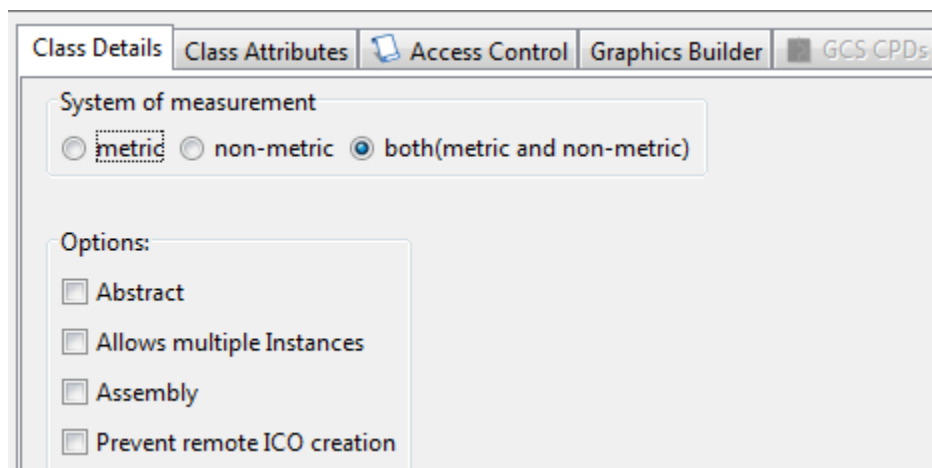
Note:

The attributes are associated with the class because they immediately follow the **SML** statement. There is no explicit reference to the class in the **SMD** statements.

Element name	Definition	Description
SML	Keyword	
SML-ID	Class or group ID	Internal unique class/group identifier.
Type	Module name	
Group	Parent class or group	For class definitions, the Group parameter defines the ID of the class or group that the current class or group belongs to in the hierarchy. You can only nest a group underneath another group. You can add a class to a group or to another class.
Description	Class name	Name displayed in the Classification user interface.
Short Description	Short name	
Graphics File	Associated image file name	
Flags	Special case flags	Flags denote the options that you set when creating a class or group. For more information, see <i>Understanding SML flags</i> .
EX1		Not used.
EX2		Not used.

Understanding SML flags

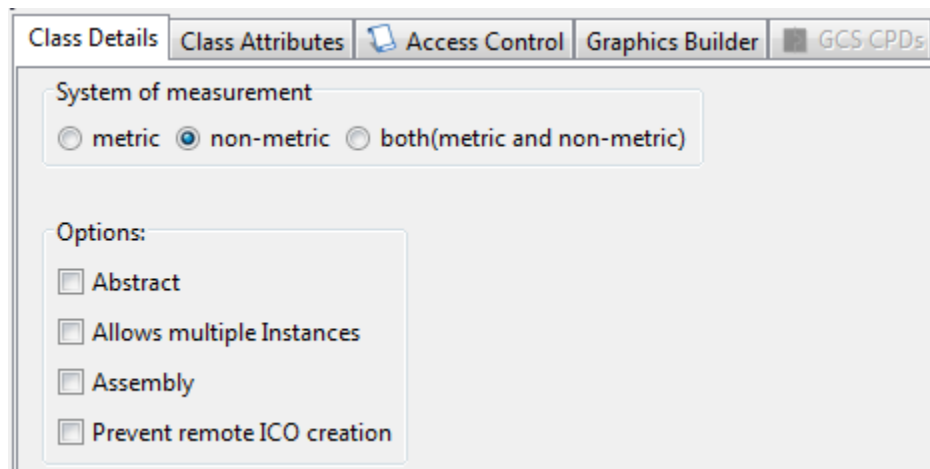
Flags denote the following options that you set when creating a class or group using the **smlutility** utility.



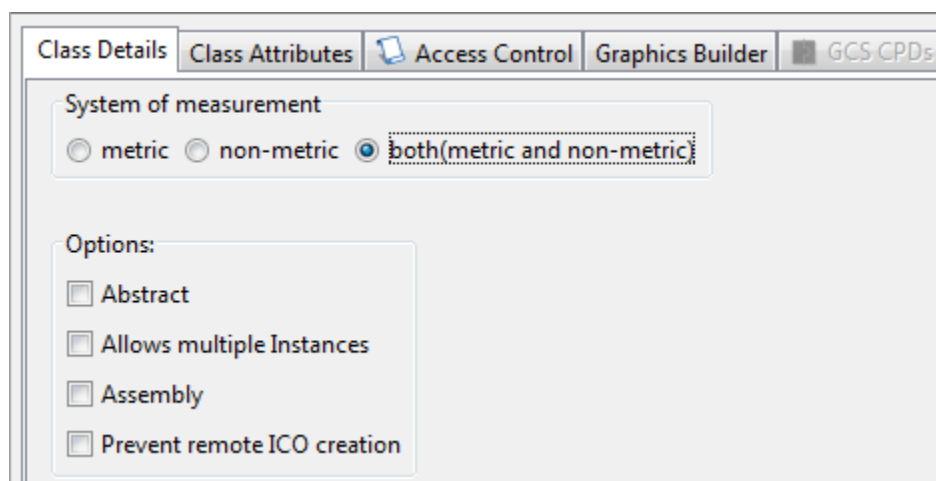
You can set these flags using specific integers in the import files.

Flag	Description
0	Class with metric unit system
1	Class with nonmetric unit system
2	Group definition
4	Class with both unit systems (metric and nonmetric)
16	Storage class (Abstract option not selected)
32	Assembly class
512	Class that prevents remote classified object creation (Prevent remote ICO creation selected)

You can also specify any meaningful combination of these options by adding the flags. For example, **17** denotes a nonmetric storage class (**16 + 1**). This is equivalent to setting the following options.



Specifying a flag of **20** denotes a storage class containing both systems of measure (**16 + 4**) and is equivalent to setting the following options.



SMD keyword (class attributes)

The **SMD** keyword associates attributes with a class. The class definition must associate all of the attributes that will be used by any subordinate subclass. A subclass can only associate attributes that have been previously associated with the parent class.

Element name	Definition	Description
SMD	Keyword	
UNCT	Unique attribute ID	Specifies the ID of an attribute defined in the UNCT dictionary. It associates the attribute with the class. For more information, see <i>SMV keyword (attributes)</i> .
Standard Designation	Standard dimensional designation DIN or ASCII standard designation for attribute	The Standard Designation and Attribute Name (from the attribute definition) appear next to the respective attribute value field in the Classification pane located on the Properties tab.
Min	Minimum numeric value	
Max	Maximum numeric value	
Flags		Not used, always 0.
EX1		Not used.
EX2		Not used.

For example, the following lines define the **Fasteners** group (essentially a class without any attributes), add the **Nuts01** class, and associate the **ID**, **OD**, and **THRD** predefined attributes with the class:

```

SML |Fasteners|FSTNRS|          |Fasteners|FSTNRS|          | 2 | |          | !
Fasteners Group
SML |Nuts01  |NT    |Fasteners|Nuts    |Nts    |nuts.gif| 0 | |          | !Nuts
Class
SMD | -25011  |ID    |          |          |0      |          |          | !ID attribute |
SMD | -25011  |OD    |          |          |0      |          |          | !OD attribute |
SMD | -25011  |Thread|          |          |0      |          |          | !Thread attribute |

```

BLD and BSM keywords (subclass)

The **BLD** keyword defines a subclass. A subclass definition consists of a **BLD** definition statement followed by up to 60 **BSM** lines that associate attributes with the subclass. The selected attributes must be a subset of the class attributes.

Note:

A subclass is similar to a view of a class. It is comprised of a subset of the attributes used to define the class. The order of the attributes does not have to be the same as the order of the attributes at the class level. Classification displays the attributes in the order defined by the subclass. The attributes are associated with the class because they immediately follow the **BLD** statement. There is no explicit reference to the class in the **BLD** statements.

Element name	Definition	Description
BLD	Keyword	
BLD-ID	Subclass ID	Internal unique subclass identifier.
Description	Description	Subclass name displayed in the Classification user interface.
Short Description	Short description	
Graphics File	Graphics file	
Flags		Not used, always 0.
EX1		Not used.
EX2		Not used.

Element name	Definition	Description
BSM	Keyword	
UNCT	Attribute ID	Specifies the ID of an attribute defined in the UNCT dictionary. It associates the attribute with the subclass. The list of attributes assigned to a subclass must be a subset of the attribute list defined for the class.

Element name	Definition	Description
Min	Minimum numeric value	
Max	Maximum numeric value	
Flags		Not used, always 0.
TXT		Not used.
Flags1		Not used, always 0.
TXT1		Not used.
Flags2		Not used, always 0.
TXT2		Not used.
EX1		Not used.
EX2		Not used.

For example, the following lines define the **HexNuts01** and **SquareNuts01** subclasses and associate a predefined attribute named ID:

```
BLD|00      |Nuts  |Fasteners|nuts.gif|Nts|nuts.gif| 0 | | |!Subclass
BLD|01      |HXNuts|Fasteners|nuts.gif|Nts|nuts.gif| 0 | | |!Subclass
BSM|-25011|ID      |          |          |0  |          |  | |!ID attribute
```

DAT keyword (instances)

The **DAT** keyword represents a Classification instance. Each **DAT** statement represents one instance.

An instance is a set of attribute values corresponding to an attribute list that defines a particular subclass. Any number of instances can exist for a specific subclass. Along with a relation to a Teamcenter object (for example, item, item revision, and dataset), the system creates a complete classification. The import/export function of the **smlutility** provides the ability to import or export Classification instances.

Element name	Definition	Description
DAT	Keyword	
DATA-ID	Instance ID	
SML ID	Class ID	
BLD ID	Subclass ID	
Flags		Not used, always 0.

Element name	Definition	Description
POM-TAG	E Refs & I Refs	
UNCT : Value	UNCT code : Value pair	Repeated for each attribute.

For example, the following statement defines the **ucg010101_001** Classification instance that belongs to the **Subclass 01** subclass of the **ugc0101** class. The 13 attribute values are entered for this subclass.

Attribute values are stated in terms of pairs of attribute ID and values separated by a colon (:) character as follows:

```
DAT | ugc010101_001 | ugc0101 | 01 | 0 | | &
| -2605: 001.500| -2603: 04| -2619: R| -2503: TMC0_00006 &
| -18032: HSS-Co5-TiN | -2653: 000.000| -2637: 050.000 &
| -2618: 006.000| -4110: 006.000| -4110: 006.000| -4100: B | -2690: 0| -2691: 1 &
| -1200: Parallel shank cutter
```

When creating a classification instance in a class that contains both metric and nonmetric units of measure, you must specify the unit of measure for the instance. Do this by adding **-630:0** for a metric instance and **-630:1** for a nonmetric instance to the attribute/value list.



12. Query utilities

bomindex_admin

Adds structured content to the search index.

SYNTAX

```
bomindex_admin [-u=user-id {-p=password | -pf=password-file} -g=group] -logfile=location_of_logfile
-bfunction=[create | delete | list | upgrade] -inputfile=location_of_inputfile
```

ARGUMENTS

-u

Specifies the user ID. This is a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-pf

Specifies the password file that holds the cleartext or encrypted password. For enhanced security, use a password file instead of the password. If the **-pf** argument is not used, the system uses the given password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-logfile

Specifies the location of the log file written by the utilities. You can specify a different location for each utility.

-function=*function-name*

Performs the following functions:

create Creates the **BOMIndexAdminData** objects based on the input file.

- delete** Finds the **BOMIndexAdminData** objects in the input file and marks them as deleted.
- list** Creates the input file for update or delete operations for existing **BOMIndexAdminData** business objects. The generated file also reports **BOMIndexAdminData** properties such as **window-uid**.
- upgrade** Upgrades the definition of BOM index tables when the property set is modified.

-inputfile

Specifies the location of a file containing the list of structure objects to index. The input file line format is as follows:

```
item-query-string | item-revision-ID | base-revision-rule | effectivity-unit |
effectivity-end-item-query-string | effectivity-date (dd-mmm-yyyy hh:mm:ss) |
variant-rules | subscribers | closure-rules
```

An example of an input file (**bomindex_admin_input.txt**):

```
item_id=HDD-0527 | B | Any Status; Working | 5 | item_id=HDD-0527 |
31-May-2013 00:00:00 | vrule1:item_id=OwnItem1:B,vrule2:,vrule3:item_id=OwnItem3:A |
MMV | clouserule1
```

- **effectivity-unit**

If you have multiple effectivity units, their numbers must be comma-separated. Also, you must repeat the effectivity end item query string for each effectivity unit, for example:

```
| 5,10,12 | item_id=HDD-0527,item_id=HDD-0527,item_id=HDD-0527 |
```

The maximum number of effectivity units you can specify is 960.

- **variant-rules**

The variant rules (also known as saved variant rules) are comma-delimited, and follow this format:

```
SVR-name:owning-item-query-string:owning-itemrevision-ID
```

The topline item revision is the default owner.

The maximum number of saved variant rules you can specify is 960.

- **subscribers**

(Optional) You may specify:

- **VDS**

Specifies that this product configuration is indexed for viewing using the Visualization Data Server. This makes the structure available faster in visualization. **VDS** requires deployment of the Visualization Data Server.

- **closure-rules**

(Optional) If a closure rule is applied for a configuration, content in the structure (excluded by the closure rule) does not appear in **where used** query results for top-level contexts.

- **structure-type**

(**Defunct**) The type of structure that the product represents. **OCC** is the only valid value.

- **owning-user**

(**Optional**) If Visualization for MMV is enabled for the product, specify the user who owns the MMV delta collection dataset.

- **MMVIdxAccessControllers**

(**Optional**) Specify the users that control access to MMV index files.

HOW TO SPECIFY OWNING USER AND MMVIDXACCESSCONTROLLERS

Each VDS site can be configured to run as a different user. **BOMIndexAdminData** table entries are returned according to the permission specified for the configured VDS user. This helps you implement export compliance using these attributes.

Example:

Site owners configured for each VDS site:

- **USA** is **VDSadmin**
- **Mexico** is **mex_VDSadmin**
- **Canada** is **can_VDSadmin**
- **China** is **chi_VDSadmin**

BOMIndexAdminData (BIAD) entries are defined as follows:

BIAD	Product	Revision Rule	Structure Type	MMV Owning User	MMV Access Users
Biad1	Ship	Latest Working	BVR	VDSadmin	mex_VDSadmin can_VDSadmin
Biad2	Truck	Latest Working	BVR	VDSadmin	mex_VDSadmin can_VDSadmin chi_VDSadmin
Biad3	Car	Latest Working	BVR	VDSadmin	chi_VDSadmin

To summarize the access for each VDS site:

- **USA** site user **VDSadmin**
Access to **Biad1**, **Biad2**, and **Biad3** because it is specified as the MMV owning user.
- **Mexico** site user **mex_VDSadmin**
Access to **Biad1** and **Biad2** only. No access to **Biad3**.
- **Canada** site user **can_VDSadmin** user
Access to **Biad1** and **Biad2** only. No access to **Biad3**.
- **China** site user **chi_VDSadmin** user
Access to **Biad2** and **Biad3** only. No access to **Biad1**.

EXAMPLE

The following command creates a search index of structures:

```
bomindex_admin -u=username -p=password -g=dba
-logfile=C:\Scratch\log\log1.txt
-function=create -inputfile=C:\Scratch\log\bomindex_admin_input.txt
```

OVERRIDING EFFECTIVITY

When you want to specify override effectivity, do not specify it in the input file containing the product configurations to index. The override effectivity in the input file is ignored during index generation, causing a discrepancy between the indexed BOM and the BOM in use.

To set override effectivity during index generation, add the effectivity data to a Revision Rule.

For example, the Revision Rule might contain:

Z_ACE_DateOverride_Rule23_10Jan2020

which includes entries for the effective date **10-Jan-2020 00:00:00**.

The corresponding input file entry for **bomindex_admin** would have this corresponding effectivity override entry:

```
item_id=ACE_KK_EC01 | A | Z_ACE_DateOverride_Rule23_10Jan2020 | | | | |
```

default_queries

With this utility, you can manage saved queries:

- Reinstall default saved queries that were deleted.
- Validate or modify saved queries.
- Migrate or delete saved queries.
- Install translated names for saved queries for a locale specified by the **-locales** option and supported by the **TCServer** machine encoding.

Siemens Digital Industries Software recommends that you run this utility as the default Teamcenter system administration user. This ensures that saved queries are protected from unauthorized modification by other users because they are owned by a user with Teamcenter administration privileges.

SYNTAX

```
default_queries [-u=user-id {-p=password | -pf=password-file} -g=group]
-locale=locale-code | ALL [-recreate] [-validate_query_name] [-validate_query_descs] [-
modify_queries=query_name(s) | ALL] [-migrate_queries=query_name(s) | ALL] [-cleanup] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If not specified, the user's default group is assumed.

-locales

Specifies the locale, using locale codes or **ALL**, for which translated query names and descriptions are installed. Specify a single locale or multiple locales in a comma-separated list, for example, **en_US,de_DE,fr_FR**. Specifying **ALL** installs all locales supported by your Teamcenter system.

Use the **preferences_manager** utility to set the **TC_language_default** environment variable as follows:

```
preferences_manager -u=Tc-admin-user -p=password -g=group
                  -mode=import -scope=SITE -preference=TC_language_default
                  -values=locale -action=OVERRIDE
```

Information about the **TC_language_default** environment variable and supported locales is in the *Teamcenter Environment Variables*. For a list of locale codes, see *Teamcenter Localization*.

-recreate

Optional parameter. Recreates the default query.

-validate_query_name

Optional parameter. Validates that the query name does not exceed the maximum length.

-validate_query_descs

Validates that the query description does not exceed the maximum length.

-modify_queries

Updates the query clauses of the specified queries with the default query clause. Specify a single query or multiple queries in a comma-separated list, for example, **ItemRevision,Item,Dataset**. Specifying **ALL** modifies all your queries.

If default saved queries have become corrupted, you can recreate them using

-modify_queries=corrupted_query_name(s).

-migrate_queries

Migrates the query clauses of the specified queries from class-based to type-based. Specify a single query or multiple queries in a comma-separated list, for example, **ItemRevision,Item,Dataset**. Specifying **ALL** modifies all your queries.

-cleanup

Deletes deprecated queries and related saved searches based on deprecated queries. Also removes deprecated queries from quick search.

-v

Specifies verbose mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

find_recently_saved_item_rev

Allows you to search for item revisions with a **UGMASTER** dataset or BOM view revision that has been modified during a range of dates. Use a date before the earliest assembly was created to ensure a listing of all changed items.

SYNTAX

```
find_recently_saved_item_rev [-u=user-id {-p=password | -pf=password-file} -g=group]
-start_date=DD-MMM-YYYY HH:MM:SS -end_date=DD-MMM-YYYY HH:MM:SS -obj_type=object-type
[-out_file=output-filename | -outItemRevKeyFile=output-filename] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-start_date

Defines the date and time from which the item revisions are searched. The time specifies a time in the current time zone for the machine where the program is running. Use the following format: *dd-mmm-yyyy hh:mm:ss*. For example, 01-Jan-2002 13:00:00.

This argument is required.

-end_date

Defines the date and time before which the item revisions are searched. The time specifies a time in the current time zone for the machine where the program is running.

Use the following format: *dd-mmm-yyyy hh:mm:ss*. For example, 01-Jan-2002 13:00:00.

If this argument is not defined, the item revisions are searched until the current date.

This argument is optional.

-obj_type

Specifies the item revision type to be searched. This argument is optional; if not defined, objects of all item revision types are searched.

-out_file

Specifies the name of the file to which the list of item revisions is sent. This argument is optional; if not defined, the output is written to the standard output.

The output includes key values.

-outItemRevKeyFile

Specifies the name of the item revision key file to which the list of item revisions is sent. This argument is optional; if not defined, the output is written to the standard output.

-h

Displays help for this utility.

ENVIRONMENT

This utility should be run from a shell where the Teamcenter environment is set.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To list the item revisions saved after January 01, 2002:

```
$TC_ROOT/bin/find_recently_saved_item_rev  
-start_date="01-Jan-2002 00:00:00" -out_file=saved_items.txt
```

find_released_item_rev

Allows you to create a query based on date and object type. The query is performed on the Teamcenter database and generates the released item revision list to identify released item revisions. Use a date before the earliest assembly was created to ensure a listing of all released items.

SYNTAX

```
find_released_item_rev [-u=user-id {-p=password | -pf=password-file} -g=group]
-start_date=DD-MMM-YYYY HH:MM:SS -end_date=DD-MMM-YYYY HH:MM:SS -obj_type=object-type
[-out_file=output-filename | -outItemRevKeyFile=output-filename] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-start_date

Defines the date and time from which the item revisions are searched. The time specifies a time in the current time zone for the machine where the program is running. This argument is required.

Use the following format: *dd-mmm-yyyy hh:mm:ss*. For example, 01-Jan-2002 13:00:00.

-end_date

Defines the date and time before which the item revisions are searched. The time specifies a time in the current time zone for the machine where the program is running.

Use the following format: *dd-mmm-yyyy hh:mm:ss*. For example, 01-Jan-2002 13:00:00.

If this argument is not defined, the item revisions are searched until the current date.

This argument is optional.

-obj_type

Specifies the object type to be searched. This argument is optional; if not defined, objects of all types are searched.

-out_file

Specifies the name of the file to which the list of item revisions is sent. This argument is optional; if not defined, the output is written to the standard output.

The output includes key values.

-outItemRevKeyFile

Specifies the name of the item revision key file to which the list of item revisions is sent. This argument is optional; if not defined, the output is written to the standard output.

-h

Displays help for this utility.

ENVIRONMENT

This utility should be run from a shell where the Teamcenter environment is set.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To list all the item revisions released after January 01, 2002:

```
$TC_ROOT/bin/find_released_item_rev -start_date="01-Jan-2002 00:00:00"  
-out_file=released_items.txt
```

query_xml

Creates, modifies, writes, deletes, and runs queries from an XML formatted file.

SYNTAX

```
query_xml [-u=user-id {-p=password | -pf=password-file} -g=group]
-v -f=xml-command-file [-o=output-file-name] [-h]
```

ARGUMENTS

-u=

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p=

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g=

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f= xml-command-file

Specifies the fully qualified name of the XML file used to control processing. This argument is mandatory.

-o= output-file-name

Specifies the name of the file to which the output from the write and execute processes are written. This argument is optional. If unspecified, the output is sent to the console. This parameter is required for the **execute** and **execute_tuple** command types.

The **-o** argument specified for outputting a generated query result file does not support appending the results of multiple queries to the specified output file. If you have more than one query specified in the **query_xml** utility, only the last query is included in the defined output file. The documentation for **-o** usage is correct for a single query but does not describe the scenario or required parameter for multiple queries and output files.

If you execute multiple queries using **query_xml**, add a parameter called **outputFileName** to the query definition to specify a unique output file for each query. Because queries can result in structurally different XML content per context, queries cannot be appended together and must be treated as unique files per query.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>

<ImanQueryCommandFile site_name="arh" site_id="id">

  <ImanQueryCommand command="execute">

    <name value="Dataset Name"/>

    <query_input_parameter name="Dataset Name" value="test1"/>

    <query_pff_post pffName="Admin - Objects By Status"
      outputFileName="d:\\outpff1.xml"/>

  </ImanQueryCommand>

  <ImanQueryCommand command="execute"

    <name value="Dataset Name"/>

    <query_input_parameter name="Dataset Name" value="test2"/>

    <query_pff_post pffName="Admin - Objects By Status"
      outputFileName="d:\\outpff2.xml"/>
```

```
</ImanQueryCommand>
```

```
<ImanQueryCommand command="execute">
```

```
  <name value="Dataset Name"/>
```

```
    <query_input_parameter name="Dataset Name" value="test3"/>
```

```
    <query_pff_post pffName="Admin - Objects By Status"
      outputFile="d:\\outpff3.xml"/>
```

```
</ImanQueryCommand>
```

```
</ImanQueryCommandFile>
```

Running **query_xml -u=Tc-admin-user -p=password -input=myfile** generates three output files (**outpff1.xml**, **outpff2.xml**, **outpff3.xml**).

-v

Specifies verbose mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*. If the **TC_TMP_DIR** variable is not set, set it to a temporary location.

FILES

As specified in *Log files produced by Teamcenter* and the following files:

- **qry_filerunner_def.dtd**

Defines the format of the driving file.

- **pffdef.dtd**

Defines the output format when the PFF option is used.

RESTRICTIONS

None.

RETURN VALUES

Return value upon success 0

Return value upon failure Nonzero

EXAMPLES

To create a query, enter the following command on the command line:

```
query_xml -f=xml-file-name -u=Tc-admin-user -p=password -o=output-file
```

- To receive output for the **execute** and **execute_tuple** command types, you must supply a **pffName** in the input XML command file and an output file name in the **-o=output-file-name** parameter on the **query_xml** call. Output file names specified in the input XML command file are ignored.
- If the query definition includes a **query_input_parameter**, where the value is a date and time, the time must be defined in Coordinated Universal Time (UTC). Teamcenter interprets the time supplied in the following example as 09:00 UTC. It does not return the time in the time zone where the program is being run.

```
<query_input_parameter name="Last Modified Date" value=
"13-Jan-2014 09:00"/>
```

XML FILE FORMAT AND EXAMPLES

The XML file must conform to the format shown in the following code segments.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ImanQueryCommandFile [
<!-- this is all we need to drive the iman query command line processor
query_xml -->
<!-- the redundant query definition -->
  <!ELEMENT name EMPTY>
  <!ATTLIST name value CDATA #REQUIRED>
  <!ELEMENT description EMPTY>
  <!ATTLIST description value CDATA #REQUIRED>
  <!ELEMENT class EMPTY>
  <!ATTLIST class value CDATA #REQUIRED>
  <!ELEMENT clauses_real (#PCDATA)>
  <!ELEMENT clauses_display (#PCDATA)>
  <!ELEMENT uniqueid EMPTY>
  <!ATTLIST uniqueid value CDATA #REQUIRED>
  <!ELEMENT iflag EMPTY>
  <!ATTLIST iflag value CDATA #REQUIRED>
  <!ELEMENT ImanQueryDefinition (name, description, class,
clauses_real,
clauses_display?, uniqueid, iflag)>
<!-- if we are executing a query - the name and value
of the search parameters... -->
  <!ELEMENT query_input_parameter EMPTY>
  <!ATTLIST query_input_parameter name CDATA #REQUIRED>
  <!ATTLIST query_input_parameter value CDATA #REQUIRED>
<!-- if we want the output of the query put through a pff and
written to a file... -->
  <!ELEMENT query_pff_post EMPTY>

  <!ATTLIST query_pff_post pffName CDATA #REQUIRED> <!-- the pff to use
(must be in db)-->
      outputFileName CDATA #REQUIRED> <!-- file to write data
(no longer used)-->
<!-- the encapsulation of the command. the attribute says it all. -->
<!-- for the create and modify the program expects the
ImanQueryDefinition -->
<!-- for the execute delete and write the name is sufficient though the--
>

<!-- full definition of the query will work. -->
  <!ELEMENT ImanQueryCommand ((name | ImanQueryDefinition),
```

```

(query_input_parameter)*,
(query_pff_post)? )>
  <!ATTLIST ImanQueryCommand  command (create | modify |
execute | execute_tuples |
delete | write_query) #REQUIRED>
<!-- the command file can contain a list of commands... -->
<!-- the site_name and site_id allow sys-admins to      -->
<!-- reconcile attribute differences based on site      -->
  <!ELEMENT ImanQueryCommandFile (ImanQueryCommand)*>
  <!ATTLIST ImanQueryCommandFile  site_name CDATA #IMPLIED
                                site_id   CDATA #IMPLIED>
]>
  <ImanQueryCommandFile site_name="fred" site_id="id">
    <ImanQueryCommand command="create">
      <ImanQueryDefinition>
        <name value="mjsABCXML_commandfilein"/>
        <description value="no description"/>
        <class value="ItemRevision"/>
        <clauses_real>
          SELECT qid FROM ItemRevision WHERE
"Form:IMAN_specification.ECOSample:data_file.charge_number"
= "${charge = }"  </clauses_real>
        <uniqueid value="0"/>
        <iflag value="0"/>
      </ImanQueryDefinition>
    </ImanQueryCommand>
    <ImanQueryCommand command="modify">
      <ImanQueryDefinition>
        <name value="mjsABC"/>
        <description value="a better description"/>
        <class value="ItemRevision"/>
        <clauses_real>
          SELECT qid FROM ItemRevision WHERE
"Form:IMAN_specification.ECOSample:data_file.charge_number"
= "${charge = }"  </clauses_real>
        <uniqueid value="0"/>
        <iflag value="0"/>
      </ImanQueryDefinition>
    </ImanQueryCommand>
    <ImanQueryCommand command="execute">
      <name value="i2ir"/>
      <query_input_parameter name="ID" value=""/>
      <query_input_parameter name="Revision" value="B"/>
      <query_pff_post      pffName="PFF Name 2" outputFileNames="
z:\\junkpff.xml"/>
    </ImanQueryCommand>
    <ImanQueryCommand command="execute_tuples">
      <name value="i2ir"/>

```

```

        <query_input_parameter name="ID" value="*" />
        <query_input_parameter name="Revision" value="B" />
    </ImanQueryCommand>
    <ImanQueryCommand command="write_query">
        <name value="i2ir" />
    </ImanQueryCommand>
    <ImanQueryCommand command="delete">
        <name value="mjsABCXML_commandfilein" />
    </ImanQueryCommand>
</ImanQueryCommandFile>

```

XML file format

```

<?xml version="1.0" encoding="UTF-8"?>
<ImanQueryCommandFile site_name="arh" site_id="id">
    <ImanQueryCommand command="create">
        <ImanQueryDefinition>
            <name value="command2file" />
            <description value="no description" />
            <class value="ItemRevision" />
            <clauses_real>
                SELECT qid FROM ItemRevision
                WHERE "object_name" = "${Name = }"
                AND    "item_revision_id" = "${Revision = }"
            </clauses_real>
            <uniqueid value="0" />
            <iflag value="0" />
        </ImanQueryDefinition>
    </ImanQueryCommand>
    <ImanQueryCommand command="execute">
        <name value="command2file" />
        <query_input_parameter name="Name" value="newnewnew" />
        <query_pff_post pffName="Admin - Objects By Status"
            outputFile="d:\\temp\\outpff4.xml" />
    </ImanQueryCommand>
    <ImanQueryCommand command="delete">
        <name value="command2file" />
    </ImanQueryCommand>
</ImanQueryCommandFile>

```

XML file example

```

<?xml version="1.0" encoding="UTF-8"?>
<ImanQueryCommandFile site_name="arh" site_id="id">
    <ImanQueryCommand command="execute">
        <name value="Item Revision..." />
    </ImanQueryCommand>
</ImanQueryCommandFile>

```

```

    <query_input_parameter name="Revision" value="A"/>
    <query_pff_post pffName="Admin - Objects By Status"
        outputFile="d:\\Users\\outpff.xml"/>
</ImanQueryCommand>
<ImanQueryCommand command="write_query">
    <name value="Item Revision..." />
</ImanQueryCommand>
</ImanQueryCommandFile>

```

XML file example

QUERY TYPES

Create, modify, and delete the following query types using the **query_xml** utility. Use the following values in the **iflag** field to specify the query type in the **.xml** file for create and modify tasks. The value for delete remains the same.

Local Query	"0"
Remote Query	"1"
User Exit Query	"8"
User Query	"16"
Structure Query	"40"

For information about where these query types are applicable, see *Search and Query Builder on Rich Client*.

The following code is an example for a BOM structure query modification.

```

<?xml version="1.0" encoding="UTF-8"?>
<ImanQueryCommandFile site_name="arh" site_id="id">
    <ImanQueryCommand command="modify">
        <ImanQueryDefinition>
            <name value="mjsABC"/>
            <description value="a better description"/>
            <class value="ItemRevision"/>
            <clauses_real>
                SELECT qid FROM ItemRevision WHERE
                    "Form:IMAN_specification.ECOSample:
data_file.charge_number"
                    = "${charge = }"    </clauses_real>
            <uniqueid value="0"/>
            <iflag value="40"/>
        </ImanQueryDefinition>
    </ImanQueryCommand>
</ImanQueryCommandFile>

```

```
</ImanQueryCommand>  
</ImanQueryCommandFile>
```

BOM structure query modification

tc_set_query_where_run

Runs a saved query. You can choose to run the saved query using a user exit.

See the `TC_ROOT\sample\examples\user_query.c` file for user exit sample code that can be used with this utility.

SYNTAX

```
tc_set_query_where_run [-u=user-id {-p=password | -pf=password-file} -g=group]
-query=query-name -run={iman | query | user} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-query

Specifies the name of saved query to run.

-run

Specifies how to run the query:

- **iman**

Run the query normally.

- **query**

Run using a user exit.

- **user**

Run using a user exit and display results in a text table.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.



13. Maintenance utilities

Installation

install

Performs limited Teamcenter maintenance and Oracle database administration.

Caution:

- This utility requires the user ID of the Teamcenter administrator account. During Teamcenter installation, this ID is **infodba**, but after installation, this must be changed to a different user account with similar privileges.

Never use the **infodba** user to create working data or initiate workflow processes. The **infodba** user is to be used *only* for certain tasks during Teamcenter installation. Using this account to create data or initiate workflow processes can cause unexpected and problematic behaviors.

If you require a user with high-level privileges to create data, create a new user and grant database administrator privileges to that user.

- This utility requires the password for the Teamcenter administrator account. The password must not be empty nor contain any whitespace characters such as space, tab, newline, carriage return, form feed, or vertical tab. In addition, the password must not contain any of the following characters:

! # @ \$ % = & ' " ^ : ; . _ < > () { }

- Some of the **install** utility arguments are reserved for Siemens Digital Industries Software use only and should not be used by customers. Using the **install** utility with arguments designated as being for Siemens Digital Industries Software use only can result in corruption of the Business Modeler IDE template and data model synchronization.

SYNTAX

install

```
{-p=password | -pf=password-file}
[-add_func_index user-id password grp class-name index-name unique-flag
fn class-name attr1 attr2...]
[-add_index user-id password group class-name index-name unique-flag
class-name attr1 attr2...]
[-ask_version]
[-ask_xmit_file user-id password group object-file]
[-drop_index user-id password group class-name index-name]
[-encrypt]
[-encryptpwf -e=environment-variable-name -f=pw-file-name]
[-find_control_chars user-id password group class-name attribute]
[-gen_xmit_file user-id password group]
[-lock_db user-id password group]
[-pom_object_index user-id password group]
[-regen_db_stats user-id password group]
[-regen_schema_file user-id password group]
[-replace_control_chars user-id password group class hex-bad-char replacement-string]
```

```

[-reset_last_login_time user-id password group]
[-revisioning user-id password group PARTITION {REPORT | APPLY | REMOVE} ]
[-set_internal_site]
[-set_pom_param [TC_TIMESTAMP_TIDY_MODE manual] [TC_TIMESTAMP_THRESHOLD
hours]
[-temp_table user-id password group list ]
[-tidy_timestamps]
[-unlock_db user-id password group]
[-mark_as_test_env user-id password group [-site_id=site-id] ]
[-check_for_test_env user-id password group [-site_id=site-id] ]
[-h]

```

ARGUMENTS

-p

Specifies the password associated with the *user-ID* value.

This argument is mutually exclusive with the **-p** argument. One of the two mutually exclusive password elements is required.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument. One of the two mutually exclusive password elements is required.

-add_func_index *user-id password grp index-name unique-flag fn class-name attr1 attr2...*

Generates an index on the **upper** field name of the given attribute to allow case-insensitive search. Other functions should only be used upon guidance from Siemens Digital Industries Software. The following values must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

grp

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

index-name

Specifies the name of the index. This is internal to POM and not the name used in Oracle.

unique-flag

1 indicates that the index is unique. **0** indicates that the index allows duplicates.

fn

If this argument is omitted, the **upper** field name is assumed.

class-name

Specifies the name of the class.

list-of-attributes

Specifies the list of attributes of the class separated by a blank space.

Note:

- The order of attributes is important. Take care when creating indexes on a group of attributes.
- The user, password, and group arguments can alternatively be supplied in one of the following forms:
 - **-u=user-id -p=password -pf=password file**
 - **-g=group**

-add_index *user-id password group index-name unique-flag class-name attr1 attr2...*

Generates a new POM schema file. Requires Teamcenter system administration privileges. The following values must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

index-name

Specifies the name of the index. This is internal to POM and not the name used in Oracle.

unique-flag

1 indicates that the index is unique. **0** indicates that the index allows duplicates.

class-name

Specifies the name of the class.

list-of-attributes

Specifies the list of attributes of the class separated by a blank space.

Note:

The order of attributes is important. Take care when creating indexes on a group of attributes.

-ask_version

Returns the current version of Teamcenter stored in the Oracle database.

-ask_xmit_file *user-id password group object-file*

Returns the **transmit file** required by a Classic Multi-Site **objects.meta** file. This requires **POM_TRANSMIT_DIR** to be set. The following arguments must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

object-file

Specifies the object file to transmit. Alternatively, use this syntax: **-f=object-file**.

Note:

- The order of attributes is important. Take care when creating indexes on a group of attributes.
- The user, password, and group arguments can alternatively be supplied in one of the following forms:
 - **-u=user-id -p=password -pf=password file**
 - **-g=group**

-drop_index *user-id password group class-name index-name*

Drops an index to revert a change made using **install -add_index**. The following arguments must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

class-name

Specifies the name of the class.

index-name

Specifies the name of the index to drop.

Note:

- The order of attributes is important. Take care when creating indexes on a group of attributes.
- The user, password, and group arguments can alternatively be supplied in one of the following forms:
 - **-u=user-id -p=password -pf=password file**
 - **-g=group**

-encrypt

Reads a database connect string from the **TC_DB_CONNECT** environment variable and displays on the console that connect string with the password encrypted.

To change the database password, you need to change the password (see Teamcenter Administration), set the **TC_DB_CONNECT** environment variable to contain that new password, run this utility specifying the **-encrypt** option, and copy the new version of the connect string it outputs into the **tc_profilevars** file.

-encryptpwf

Creates an encrypted password file. Requires the **-e** argument to identify the environment variable that contains the password to encrypt and the **-f** argument that indicates the name and location of the file containing the encrypted password.

-e= *environment-variable-name*

Specifies the environment variable that contains the password to encrypt. This argument is valid only if you specify the **-encryptpwf** argument. You can specify any environment variable that contains a clear text password value that you want to encrypt. For security reasons, choose an obscure and unique name for the environment variable and immediately unset the environment variable after you run the utility.

-f= *password-file-name*

Specifies the name and location for the file containing the encrypted password used to connect to the Teamcenter database. This file is used by Teamcenter utilities when using the **-pf** argument and by some other Teamcenter functions. This argument is required when you use the **-encryptpwf** argument and is valid only if you specify the **-encryptpwf** argument.

-find_control_chars *user-id password group class-name attribute*

Finds control characters in the Teamcenter database. Control characters can enter the Teamcenter database through ITK operations that are legitimate codes for the server or database code pages, but cause problems for XML because they are illegal characters and thus cause problems for SOA based integrations in particular. To find such characters, the following arguments must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

class-name

Specifies the name of the class.

attribute

Specifies the attribute.

Note:

- The order of attributes is important. Take care when creating indexes on a group of attributes.
- The user, password, and group arguments can alternatively be supplied in one of the following forms:
 - **-u=user-id -p=password -pf=password file**

- **-g=group**

-gen_xmit_file *user-id password group*

Generates the transmit file containing a copy of the Teamcenter schema that is used by POM during import to compare the exporting site schema definition to the importing site schema definition for all classes. The file resides in the **\$POM_TRANSMIT_DIR** directory.

Note:

In some cases, the **POM_TRANSMIT_DIR** files must be deleted and regenerated when they fail to update during the **-gen_xmit_file** command.

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-lock_db

Locks the sites against further use. The lock remains in place until unlocked with the **-unlock_db** argument. The following values must be supplied in this order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

See restrictions #1 and #4.

-pom_object_index *user-id password group*

Specifies an optional index on the **POM_object** class that allows queries to answer high frequency queries directly from the indexes (instead of referring to the table row). This is not a default argument because it requires additional database server memory.

This affects the following indexes:

- **PPOM_OBJECT_SLN** on **puid** and **ptimestamp**
- **PPOM_OBJECT_UPI** on **puid** and **ppid**

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-regen_db_stats

Gathers database statistics for the schema and/or specific tables within the database. It provides administrators an efficient alternative to querying the database directly. This argument operates on Oracle databases only.

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

all

Regenerates table statistics for all tables (forced regeneration). When this parameter is used alone, the utility queries the tables and then displays all statistics to the console window, including stale or empty statistics tables.

sql

Displays to the console window the **gather_schema_stats** sql query that would gather the statistics. It does not perform the actual gathering of statistics.

templates

Displays to the console window the **gather_table_stats** sql query that would gather table statistics for the template. It does not perform the actual gathering of statistics.

noreport

Suppresses much of the output to the console window. When used with the *all* option, it omits tables with stale or empty statistics.

table1, ...

Narrows the scope of any of the options above to a specific table or list of tables to act upon.

-regen_schema_file

Generates a new POM schema file. Requires Teamcenter system administration privileges. The following values must be supplied in order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**. See restrictions #1 and #3.

-replace_control_chars *user-id password group class hex-bad-char replacement-string*

Replaces control characters found by the **-find_control_chars** argument.

An temporary alternative to fixing the data in the database is to set an environment variable (in the *TC_DATA\tcprofilevars.bat* file) to replace control characters. For example, to replace control characters with spaces (ASCII character 32), set the following variable:

POM_STRIP_CTRL_CHARS=32

To use the **-replace_control_chars** argument, the following values must be supplied in this order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

class

Specifies the class.

hex-bad-char

Specifies the hexadecimal code of the control character to replace.

Do *not* prefix this value with **0x** to indicate a hexadecimal value, enter only the hexadecimal value itself. For example, a value of **1f** is valid, but **0x1f** is not valid.

replacement-string

Specifies the string to insert in place of control characters.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-reset_last_login_time *user-id password group*

Updates the last login time for the user authenticating through a command line. This argument is suitable for batch processing. The following values must be supplied in this order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-reversioning user-id password group PARTITION {REPORT | APPLY | REMOVE

POM-revisable tables (used by 4GD) use database partitioning for historical data automatically during upgrade or install based on the feature availability with the database. Partitioning helps query performance where a large volume of historical data is present.

Database partitioning capability is available in Oracle Enterprise, but in Teamcenter it is controlled by the **TC_USE_REV_PARTITIONING** environment variable.

By default, this environment variable is set to **N** because database partitioning requires a license at extra cost from Oracle. If you want to enable partitioning for 4GD data, set **TC_USE_REV_PARTITIONING** to **Y** before installation or upgrade, or execute the **install** utility with the **-reversioning PARTITION** argument.

Following are the parameters that can be used with the **-reversioning PARTITION** argument:

- **REPORT**

Report about classes supporting minor revisions which use tables that are not partitioned.

- **APPLY**

Apply database partitioning to tables of classes supporting minor revisions.

- **REMOVE**

Remove database partitioning from tables of classes supporting minor revisions.

-set_internal_site

Sets an internal site ID. This is run automatically when using the **Create environment for upgrade testing** check box in Teamcenter Environment Manager (TEM). Setting the internal site ID allows FMS to communicate with both the test environment and the original production environment.

-set_pom_param

Sets a POM parameter. Use in combination with the following parameters:

TC_TIMESTAMP_THRESHOLD

Sets the threshold for the number of hours to hold timestamps in the **POM_timestamps** table. The default is 96 hours. Use the following format to change the threshold:

```
install -set_pom_param -u=user -p=password -g=group TC_TIMESTAMP_THRESHOLD hours
```

To reset the threshold back to the default threshold, rerun the command without the *hours* variable.

TC_TIMESTAMP_TIDY_MODE

Allows manual cleaning of the **POM_TIMESTAMP** table. Use the following format:

```
install -set_pom_param -u=user -p=password -g=group TC_TIMESTAMP_TIDY_MODE manual
```

After enabling manual cleaning, run the **-tidy_timestamps** argument to remove timestamps from the **POM_TIMESTAMP** table. To disable manual mode, rerun the parameter without the **manual** variable.

-temp_table *user-id password group list*

[*list* | *drop* [*all* | *older_than_date=YYYY/MM/DD HH:MI:SS* | *table_name=temp-table-name*]]

Lists Oracle temporary table names and drops them if necessary. You can use this utility to check whether Oracle temporary table definitions are accumulating and drop them to clean them up.

Oracle temporary table definitions can accumulate when the server exits abruptly, for example, when a database connection is lost.

You can list these table definitions using the **list** option or purge them using the **drop** option with a specific name or date, or purge all.

The following values must be supplied in this order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

list

Specifies to list accumulated temporary table definitions.

Alternatively, use the **drop** option instead of **list** to drop temporary table definitions.

Note:

Although Oracle prevents database users from dropping temporary tables that are in use by running processes, it is best to drop temporary tables when the system is quiet, when no users are logged in.

- **drop all**

Drops all temporary table definitions.

- **drop older_than_date=YYYYMMDD HH:MI:SS**

Drops definitions older than the specified date and time.

- **drop table_name=temp-table-name**

Drops the specified temporary table.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-tidy_timestamps

Removes timestamps from the **POM_timestamps** table. Each timestamp records the time of an object's most recent modification.

The **POM_timestamp** table holds timestamps for a configured amount of time. (The default is 96 hours.) By default, Teamcenter performs maintenance on the **POM_timestamp** table at session logout. If this becomes a bottleneck due to the volume of concurrent session logouts, moving to manual maintenance offers better control. Before using the **-tidy_timestamps** argument to clean out timestamps, you must enable manual mode by using the **-set_pom_param** argument with the **TC_TIMESTAMP_TIDY_MODE manual** parameter.

-unlock_db user-id password group

Releases locks set with the **-lock_db** argument. The following values must be supplied in this order:

user-id

Specifies the user ID of the Teamcenter administrator account.

password

Specifies the password or password file associated with the *user-ID* value.

For more information, see the descriptions of the **-p** and **-pf** arguments.

group

Specifies the group associated with the *user-ID* value. In most cases, the group is **dba**.

See restriction #1.

-mark_as_test_env

Changes a production environment to a test environment that can use the bulk load features for copying product data.

Caution:

Because you cannot revert a test environment to a production environment, you are prompted to confirm this action.

You can specify the **-site_id=** argument to change a remote site to test environment in the local site database. This selects the read-only **Is A Test Environment** check box for the site in the Organization application to show that it is a test environment. If you do not supply a site ID, the utility prompts you to confirm the change to the local environment.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u=user-id -p=password -pf=password file**
- **-g=group**

-check_for_test_env

Checks the environment to determine whether it is a test or production environment. This can be used before attempting to change an environment to a test environment and to verify that an production environment has been changed to a test environment. You can specify the **-site_id=** argument to check other sites known to the local environment. If you do not supply a site ID, the utility returns the status of the local site.

Note:

The user, password, and group arguments can alternatively be supplied in one of the following forms:

- **-u**=*user-id* **-p**=*password* **-pf**=*password file*
- **-g**=*group*

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#) and the following environment variables:

TC_DB_CONNECT
POM_SCHEMA
POM_TRANSMIT_DIR

For more information, see the *Teamcenter Environment Variables*.

FILES

As specified in [Log files produced by Teamcenter](#) and the following:

- **\$TC_DATA/tc_profilevars**

Stores site environment variable settings. This file is modified by the **-encrypt** argument.

- POM schema file data file created by the **install** utility with the **-regen_schema_file** argument.

Full file specification (directory path and file name) is set by the **POM_SCHEMA** environment variable.

- POM transmit schema file data file created by the **install** utility with the **-gen_xmit_file** argument.

Full file specification (directory path and file name) is set by the **POM_TRANSMIT_DIR** environment variable.

RESTRICTIONS

1. Common command line argument syntax for *user-id*, *password*, and *group* arguments is not supported. Values for these arguments must be separated by an equal sign (=). For example, the following syntax works:

```
$TC_BIN/install -regen_schema_file tcAdmin password dba
$TC_BIN/install -regen_schema_file -u=tcAdmin -p=password -g=dba
```


2. Requires Teamcenter system administration privileges and exclusive access to the system for this operation.

3. Common **-regen_schema_file** failures:

POM_db_connect_fail

Unable to connect to database.

POM_logins_are_disabled

Login to database is disabled.

POM_invalid_site_id

Database is not populated.

POM_not_installed

Database missing data.

POM_find_schema_failed

Unable to create new POM schema file. Directory does not exist, cannot be written to, or the **POM_SCHEMA** environment variable is not set.

POM_schema_exists

File pointed to by the **POM_SCHEMA** environment variable already exists. Delete or move this file and retry.

4. The **-lock_db** does not force logout of existing users, but does prevent additional users from logging on.

5. Only the following tokens can be changed on an existing (saved class):

```
POM_attr_export_as_string
POM_attr_follow_on_export
POM_attr_is_candidate_key
POM_null_is_valid
POM_public
POM_public_read
POM_public_write
```

For additional information, see *Teamcenter Server Customization*.

6. When adding a new custom privilege, you must have previously added that privilege to the **am_text.uil** file and recompiled the file.

For additional information, see *Teamcenter Server Customization*.

7. Rules-based object protection must be enabled in order to add and use new custom privileges.

EXAMPLES

- To regenerate the POM transmit schema file, enter the following command on a single line:

```
$TCROOT/bin/install -gen_xmit_file tcAdmin password dba
```

- To change a production environment to a test environment:

```
$TCROOT/bin/install -mark_as_test_env tcAdmin password dba
```

tem

Runs the Teamcenter Environment Manager utility from a command line so that it can be used to install features or update templates without user interaction with the Teamcenter Environment Manager (TEM) graphical user interface. This utility can only be run from the *application_root\install* directory.

Note:

The **tem.sh/.bat** file runs the Teamcenter Environment Manager (TEM) graphical user interface.

For information about how to use the graphical user interface, see *Teamcenter Environment Manager — Usage*.

SYNTAX

tem

```
-pass=password

-ospass=
-dbpass
-install [-features=guid,template-or-data-model-name -path=directory] -config=configuration-ID
-update [-templates=template-name1,template-name2,... {-full | -live} ] -config=configuration-ID
-path=path1,path2,... -dryrun
-minikit=target-directory
-encrypt=string -legacy
-guid
-gatewayurl=Gateway-URL
-patch=patch-location
-publish | -nopublish
-restart -config=config-ID
-s -silent=silent-file
-show -app=app-ID
-sysinfo
-help
```

ARGUMENTS

```
-pass=password
    Specifies the Teamcenter administrative user password.

-ospass=password
    Specifies the operating system user password.

-dbpass=password
    Specifies the database user password.

-install
```

Installs a feature. You must specify the following:

- **-features=***guid,template-or-data-model-name*

Specifies the features to install.

- **-path=***directory*

Specifies the path to the feature files.

- (Optional) **-config=***configuration-ID*

Specifies the configuration to install to in case there are multiple configurations available.

Note:

Not all features can be installed or updated using the command line utility, however. Only those features that are data models or have subfeatures with data models apply. For example, the rich client feature cannot be installed or updated using the tem command line utility. But the Teamcenter Automotive Edition feature can.

-update

Updates the data model. You must specify the following:

- **-templates=***template-name1,template-name2,...*

Specifies the templates to update.

- **{-full | -live}**

Use the **-full** argument to specify a full update or **-live** to specify a live update.

- **-path=***path1,path2,...*

Specifies paths to the templates to update.

- (Optional) **-config=***configuration-ID*

Specifies the configuration to update in case there are multiple configurations available.

Note:

Not all features can be installed or updated using the command line utility, however. Only those features that are data models or have subfeatures with data models apply. For example,

the rich client feature cannot be installed or updated using the **tem** command line utility, but the Teamcenter Automotive Edition feature can.

-dryrun

Tests whether the data model can be integrated successfully. This flag validates that all the needed files are present and viable and checks that the data model is in order, but it does not validate that the data model changes can actually be applied to the database. Certain contexts (like display rules depending on particular groups and roles) are not validated and may fail if the needed groups do not exist.

-encrypt=string

Encrypts a string.

-gatewayurl=URL

Specifies the URL to the Active Workspace Gateway, for example: **http://myhost:3000**

This argument must be specified when patching Active Workspace silently.

-guid

Generates a new GUID.

-minikit=target-directory

Creates a small deployable package.

-nopublish

Does *not* publish Active Workspace content to the File Repository.

This argument is mutually exclusive with the **-publish** argument.

The **-nopublish** or **-publish** argument must be specified when patching Active Workspace silently.

-publish

Publishes Active Workspace content to the File Repository.

This argument is mutually exclusive with the **-nopublish** argument. Use this argument with the **-gatewayurl=** argument.

The **-nopublish** or **-publish** argument must be specified when patching Active Workspace silently.

-patch

Specifies the location of the Teamcenter patch to apply.

-restart

Restarts a failed operation.

-s

Performs a silent installation. Use the **-s** argument to bypass the Teamcenter Environment Manager user interface and run the installation in the background. There is no feedback when the silent

installation is running. Teamcenter Environment Manager (TEM) searches the current working directory for the configuration file to use. If a file name is specified for this argument, TEM uses the specified file as input for the silent installation.

-show

Shows information. You can specify the following:

- **-app=app-ID**

Specifies the ID of the application.

- **-features**

Lists features available on the media.

- **-configs**

Lists configurations within an installation.

- **-installed**

Lists installed features.

- **-config=configuration-ID**

Specifies the configuration to use.

-sysinfo

Displays system information.

-help

Displays help text.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To install a feature, use the **-install** argument, for example:

```
tem -install -features=feature-name -path=location-of-template-files -p=password -verbose
```

- To perform a full model update, use the **-update** argument with the **-full** flag, for example:

```
tem -update -full -templates=template-name-1,template-name-2 -path=path1,path2 -pass=password
```

- To perform a live update, use the **-update** argument with the **-live** flag, for example:

```
tem -update -live -templates=template-name-1,template-name-2 -path=location-of-template-files  
-pass=password
```

- To install or update in dry run mode to validate the process before committing the changes to the database, use the **-dryrun** argument, for example:

```
tem -install -dryrun -features=feature-name -path=location-of-template-files -pass=password
```

Or:

```
tem -update -full -dryrun -templates=template-name -path=location-of-template-files  
-pass=password
```

- To create a small deployable package, use the following command:

```
tem -minikit=target-directory -silent=slient-file -config=configuration-ID
```

warPackager

The Web Application Manager (**insweb**) uses a **warPackager** utility to create a WAR file from files in the staging directory and regenerate the **application_map.xml**, **system_preferences.xml**, and **web.xml** files.

The following table lists the command flags and options.

Flag	Option	Description
-a	No option	Generates the master application map file (application_map.xml) in the location specified by the -d option.
-d	<i>staging-area</i>	Specifies the staging area path for the Web application. This option is required whenever you run the utility. If -d is the only option passed into the warPackager utility, there must be solutions.dat , webapp.dat , and context_params.dat files in the staging area directory, or the utility errors out. If those files exist, then the utility generates all files (the same as choosing -apxw).

Flag	Option	Description
-l	<i>icd-directory</i>	The directory under which the ICD directory is located.
-p	No option	Generates the master system preferences file (system_preferences.xml) in the location specified by the -d option.
-t	No option	Generates template webapp.dat and context_params.dat files in the location specified by the -d option. It is used only in conjunction with the -d option. If the -t option is used with any other option, then the utility errors out. When the -t and -d options are used together, only a solutions.dat file must exist in the staging area. The utility then generates only the webapp.dat and context_params.dat template files. If either of these template files already exist in the staging area, the utility prompts users asking if they want to overwrite these files.
-w	<i>No option</i>	Generates the WAR file in the location specified by the -d option (using the value given in the webapp.dat file for the [WAR_NAME] tag).
-x	No option	Generates the web.xml file in the location specified by the -d option.

Optionally, you can run the **warPackager** utility directly by performing the following steps:

1. Set the **CLASSPATH** environment variable to include the *WEB_ROOT/java/classes* directory.
2. Navigate to the *WEB_ROOT* directory.
3. Enter the following command:¹

```
warPackager -d deployment-directory [-i icd-directory] [-t] | [-apxw]
```

The following examples show how the **warPackager** utility can be used with its various options:

- Generate all files into the **scratch/webroot/mywebapp** directory:

```
warPackager -d /scratch/webroot/mywebapp
```

or

```
warPackager -d /scratch/webroot/mywebapp -apxw
```

- Generate template files into the **scratch/webroot/mywebapp** directory:

```
warPackager -d /scratch/webroot/mywebapp -t
```

- Generate only the **web.xml** file and the WAR file in the **/scratch/webroot/mywebapp** directory:

```
warPackager -d /scratch/webroot/mywebapp -xw
```

- Generate only the master application map and system preferences files in the **/scratch/webroot/mywebapp** directory:

```
warPackager -d /scratch/webroot/mywebapp -ap
```

Deployment Center

generateMiniSoftware

Generates a mini software kit to be used with mass client deploy scripts generated by Deployment Center.

¹ Linux systems require you to use an uppercase **P** when you enter the **warPackager** command.

generateMiniSoftware uses information in mass client deploy scripts to find and collect the files needed for the client deployment. It then builds a new software kit that contains *only* the files necessary for that deployment. The resulting mini software kit is smaller than the original software kit.

The utility can gather files from multiple software kits (for example, Teamcenter Foundation, Active Workspace, and Microservice Framework) but generates a single mini software kit for each mass client deploy script.

A mass client deploy script can perform an installation or an update of a client.

Syntax

generateMiniSoftware

[-deployScriptDir=deploy-script-dir] [-softwareLocation=software-dir] [-outDir=output-dir] [-includeAllConfiguredApps]

Arguments

-deployScriptDir *deploy-script-dir*

Specifies the path to the mass client deployment scripts directory. Be sure the scripts are in .zip file format.

-softwareLocation *software-dir*

Specifies the path to one or more full software kits from which the utility gathers files to assemble the mini software kit. The utility accepts the following values for this argument:

default	Use this value to gather software files from software repositories registered with Deployment Center.
<i>path1, ...</i>	Enter one or more paths to registered file repositories or unregistered directories that contain the software kits from which to gather files. Separate multiple paths with commas. Make sure software kits in these locations are expanded (unzipped).

-outDir *output-dir*

Specifies the location in which to generate the mini software kit. Make sure this location is writable.

The output mini-kit files are named in the following format:

env_name_mass_client_instance_OS.zip

-includeAllConfiguredApps

(Optional) Generates a mini kit that can be used to install a rich client or update an existing rich client. Specifying this argument ensures that all applications listed in the mass deploy script are copied into the mini software kit.

To generate a mini kit to be used *only* for updating an existing rich client, omit the **includeAllConfiguredApps** argument. This results in a smaller mini kit, but it cannot be used to install a new rich client.

Environment

The *generateMiniSoftware* utility is in the *minisoftwaregenerator.zip* file in the *additional_tools* directory on your Deployment Center machine. After you install or upgrade Deployment Center, locate this zip file and expand it to find the Windows and Linux versions of the *generateMiniSoftware* utility.

When you run *generateMiniSoftware* for a specified directory, it generates one mini-kit deployment script for each mass client deployment script in that location. Run *generateMiniSoftware* as an administrator.

Files

This utility generates the following files:

- Output mini-kit files, which are named with the following format:

env_name_mass_client_instance_OS.zip

- Output log file

Examples

- To generate a mini software kit that supports updating an existing rich client, enter the following command in a single line:

```
generateMiniSoftware.bat -deployScriptDir=D:\DC_deploy_scripts
                        -softwareLocation=E:\DC_software -outDir=D:\mini_deploy_scripts
```

- To generate a mini software kit that supports updating an existing rich client or installing a new rich client, enter the following command in a single line:

```
generateMiniSoftware.bat -deployScriptDir=D:\DC_deploy_scripts
                        -softwareLocation=E:\DC_software -outDir=D:\mini_deploy_scripts
                        -includeAllConfiguredApps
```

- To generate a mini software kit that supports updating or installing and draws software from registered software repositories, enter the following command in a single line:

```
generateMiniSoftware.bat -deployScriptDir=D:\DC_deploy_scripts
                        -softwareLocation=default -outDir=D:\mini_deploy_scripts
                        -includeAllConfiguredApps
```

The utility generates an output log file. After you run the utility, the output log file contains information about which components and applications are included in the mini-kit as well as information about the size reduction from the original software kits.

For information about deploying a mini-kit, see *Deployment Center — Usage*.

Audit Manager

audit_purge

Archives audit logs in TC XML format (based on retention period and audit log type) and purges audit log records. In addition, this utility supports archive and restore of audit logs:

- Use the **nearline_archive** argument to move the production data to the archive site.
- Use the **nearline_restore** argument to move the archive data from the archive site to the production site.

SYNTAX

```
audit_purge [-u=user-id {-p=password | -pf=password-file} -g=group]
[-logtype=audit-log-business-object-name]
{-purge | -archive}
[-sublogtype=process_audit | signoff_audit]
[-retentionperiod=retention-period-of-audit-log-business-object]
[-archivelocation=path-of-archive-location]
[-nearline_archive]
[-nearline_restore]
[-maxlimit_nearline_archive_restore=max-audit-logs]
[-batchsize_nearline_archive_restore=batch-size]
[-dryrun]
[-report_file]
[-verbose]
[-force_retraverse]
[-max_limit=5000]
[-opt_skip_exp_obj_allprops]
[-objecttype=Item,Text]
[-eventtype=__Check_In,__Check_Out]
[-created_before=date, time]
[-created_after=date, time]
[-groupname=dba, Engineering]
[-owner=Tc_admin]
[-skip_object_type=Dataset,Part]
{-skipInProcessWorkflowAudit}
[-objectid=]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument cannot be replaced with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see the *Teamcenter Utilities*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-logtype

Specifies the name of the audit log business object. For example, **Fnd0WorkflowAudit**, **Fnd0ScheduleAudit**.

Note:

If the audit log is **Fnd0WorkflowAudit**, you must also use the **-sublogtype** argument.

-purge

Purges the specified audit logs

If you do not also specify the **-retentionperiod** argument, Teamcenter selects the retention period from the **Fnd0RetentionPeriod** business constant.

-archive

Archives the audit log in TC XML format.

If you do not also specify the **-archivelocation** argument, Teamcenter uses the archive location specified in the **Fnd0ArchiveLocation** constant.

-sublogtype

Specifies the sublog types of the **Fnd0WorkflowAudit** object. The sublog types of the **Fnd0WorkflowAudit** object are **process_audit** and **signoff_audit**.

-retentionperiod

Specifies the retention period in days. For example, a value of **90** will purge/archive audit entries more than 90 days old. A value of **0** (zero) will not retain any log data; it will purge/archive the entire log.

The **-retentionperiod** argument overrides the retention period specified in the **Fnd0RetentionPeriod** constant.

Note:

-retentionperiod cannot be used with **-created_before=** or **-created_after=** options.

-nearline_archive

Archives audit records from production site to archive site and purges audit records from production site. This archives in-process workflow audit logs. Use the **-skiplnProcessWorkflowAudit** argument to retain in-process audit logs at production site.

Note:

Nearline archived audit records data can be restored from production site with **-nearline_restore** argument.

-nearline_restore

Restores audit records from archive site to production site and purges audit records from archive site.

-maxlimit_nearline_archive_restore

Indicates the maximum number of audit logs allowed for nearline archive and nearline restore per execution.

The default limit is 1000000.

Note:

This option should be used only when **-nearline_archive** or **-nearline_restore** argument is provided.

-batchsize_nearline_archive_restore

Indicates the number of audit logs to be processed per batch. The audit logs are processed in multiple batches based on the total number of audit logs found and the given batch size.

The default batch size is 50000.

Note:

This option should be used only when **-nearline_archive** or **-nearline_restore** argument is provided.

-dryrun

Performs dry run and reports the objects that would be archived or restored from production site.

Note:

This option should be used only when **-nearline_archive** or **-nearline_restore** argument is provided.

-report_file

Specifies that a report be output to the given file path.

Note:

This option should be used only when **-nearline_archive** or **-nearline_restore** argument is provided.

-verbose

Runs in **Verbose** mode to show detailed messages in the utility's output.

Note:

This option should be used only when **-nearline_archive** or **-nearline_restore** argument is provided.

-archivelocation

Specifies the path of the archive location.

The **-archivelocation** argument overrides the value of archive location specified in the **Fnd0ArchiveLocation** constant.

-force_retraverse

Re-archives audit logs. Use this argument with the **-archive** argument to archive the same objects again.

-max_limit

Specifies the maximum number of audit logs to archive per **xml** file.

For example, filename_1.xml [1-5000 records],

filename_2.xml [5001-10000 records]

-opt_skip_exp_obj_allprops

Provides the option to purge or archive audit records based on a specified object type.

-objecttype=Item,Text

Specifies the object type.

Provides the option to purge or archive audit records based on a specified event object type.

-eventtype=__Check_In,__Check_Out

Specifies the event type.

Provides the option to purge or archive audit records based on a specified event type.

-created_before=date, time

Specifies the created before date and time.

Provides the option to purge or archive audit records created before a specified date.

-created_after=date, time

Specifies the created after date and time.

Provides the option to purge or archive audit records created after a specified date.

Note:

-retentionperiod cannot be used with **-created_before=** or **-created_after=** options.

-groupname=dba, Engineering

Specifies the group name.

Provides the option to purge or archive audit records based on a specified group.

-owner=Tc_admin

Specifies the User id.

Provides the option to purge or archive audit records based on a specified owner.

-skip_object_type=Dataset,Part

Specifies the object type to skip.

Provides the option to skip the object type during purge.

Note:

-skip_objecttype and **-objecttype** cannot be used together.

-skipInProgressWorkflowAudit

Provides the option to archive or purge audit records for completed and aborted workflows.

Note:

This should be used when **-logtype** is **Fnd0WorkflowAudit**.

-objectid=

Provides the option to purge or archive audit records based on a specified object.

Note:

Value of object id should be Primary Object ID for Audit record.

-h

Displays help for this utility.

RESTRICTIONS

Requires Teamcenter administrator privileges.

EXAMPLES

Note:

When archiving or purging the **Fnd0WorkflowAudit** audit log, you must use the **-sublogtype** argument with the **-logtype** argument.

The sublog types of the **Fnd0WorkflowAudit** audit log are **process_audit** and **signoff_audit**.

- To purge the signoff audit logs of the **Fnd0WorkflowAudit** signoff audit log:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0WorkflowAudit -sublogtype=signoff_audit -purge
-retentionperiod=90
```

- To purge the process audit logs of the **Fnd0WorkflowAudit** signoff audit log with a retention period of 90 days:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0WorkflowAudit -sublogtype=process_audit -purge
-retentionperiod=90
```

Note:

Data more than 90 days old will be purged. Data generated within 90 days will be not be purged.

- To archive the process audit logs of the **Fnd0WorkflowAudit** audit log:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0WorkflowAudit -sublogtype=process_audit
-archive -retentionperiod=90 -archivelocation=c:\archive
```

- To archive the process audit logs of the **Fnd0WorkflowAudit** audit log with a retention period 90 days:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0WorkflowAudit -sublogtype=process_audit
-archive -retentionperiod=90 -archivelocation=c:\archive
```

- To archive the process audit logs of the **Fnd0WorkflowAudit** audit log with the archive location specified as **c:\archive**:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0WorkflowAudit -sublogtype=process_audit
-archive -retentionperiod=90 -archivelocation=c:\archive
```

- To purge audit records based on persistence object model object types:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-objecttype=Document -purge -retentionperiod=90
```

- To purge or archive audit records based on a check in event:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-eventtype=__Check_In -purge -retentionperiod=90
```

- To purge audit records created before June 16, 2018 at 4pm:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-created_before="16-Jun-2018 16:00" -purge
```

- To purge audit records created after June 16, 2018 at 4pm:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-created_after="16-Jun-2018 16:00" -purge
```

- To purge audit records specific to the **Engineering** group:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-groupname=Engineering -purge -retentionperiod=90
```

- To purge audit records owned by a user named Paul Programmer:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-owner=Paul -purge -retentionperiod=90
```

- To skip persistence object model object types during purge:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-skip_object_type=Document -purge -retentionperiod=90
```

- To archive audit records data from production site to the archive site for specified time period of 30 days:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
```

```
-created_after="01-Jun-2018 00:00" -created_before="30-Jun-2018 00:00"
-nearline_archive
```

- To restore the general audit logs of object of having id as **200012** from the archive site to the production site keeping latest 90 day's audit records data:

```
audit_purge -u=Tc-admin-user -p=password -g=group
-logtype=Fnd0GeneralAudit
-objectid=200012 -retentionperiod=90 -nearline_restore
```

- To archive 1100000 audit logs, set the maximum limit to 1100000.

Note:

If you do not specify the maximum limit argument, you will not be able to archive audit logs exceeding 1000000 for the given input arguments.

```
audit_purge -u=Tc-admin-user -p=password -g=group -logtype=Fnd0GeneralAudit
-created_after="09-Mar-2020 00:15"
-created_before="30-Mar-2020 00:16" -verbose -nearline_archive
-maxlimit_nearline_archive_restore=1100000
```

- To restore 5000 audit logs in two batches, you can set the batch size to 3000:

Note:

If you do not specify the batch size argument in this example, all 5000 audit logs are restored in a single batch.

```
audit_purge -u=Tc-admin-user p=password -g=group -logtype=Fnd0GeneralAudit
-created_after="09-Mar-2020 00:15"
-created_before="09-Mar-2020 00:16" -verbose -nearline_archive
-batchsize_nearline_archive_restore=3000
```

audit_archive

Note:

This utility is deprecated and will be removed in a future version.

Searches the database for audit log records based on input criteria. Once it finds the records that must be archived, it processes the archive. If the **-delete_record** argument is given, the utility deletes the audit log records. Audit log entries with an audit definition with a **days kept** value of **-1** are not archived, because **-1** indicates that the log record is permanent.

SYNTAX

```
audit_archive [-u=user-id {-p=password | -pf=password-file} -g=group]
[-delete_record] [-type=type-name] [-class=class-name] [-event=event-type]
[-id=object-id]
[-revid=object-rev-id] [-name=object-name] [-owner=object-owner-id]
[-created_before=archive-date] [-created_after=archive-date]
[-group_name=owner-group-name]
[-obj_seqno=object-sequence-number]
[-secobj_type=secondary-object-type-name]
[-secobj_id=secondary-object-id]
[-secobj_name=secondary-object-name]
[-secobj_revid=secondary-object-rev-id]
[-secobj_seqno=secondary-object-sequence-no]
[-error_code=error-code-of-failed-action]
[-proj_id=project-id] [-proj_name=project-name]
[-overwrite] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument cannot be replaced with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-delete_record

Specifies that audit log records are deleted after being archived.

-type

Specifies that the audit logs of the specified object type needs to be archived.

-class

Specifies that the audit logs of the specified class of the object type needs to be archived.

-event

Specifies that the audit logs of the specified event type needs to be archived.

-id

Specifies that the audit logs of the specified ID of the object needs to be archived.

-revid

Specifies that the audit logs of the specified revision ID of the object needs to be archived.

-name

Specifies that the audit logs of the specified name of the object needs to be archived.

-owner

Specifies that the audit logs of the specified user who created the object needs to be archived.

-created_before

Specifies that the audit logs of objects created before the specified date need to be archived.

-created_after

Specifies that audit logs of objects created after the specified date need to be archived.

-group_name

Specifies that the audit logs of the specified group name associated with the user who created the object needs to be archived.

-obj_seqno

Specifies that the audit logs of the specified sequence number of the object needs to be archived.

-secobj_type

Specifies that the audit logs of the specified type of the secondary object associated with the object needs to be archived.

-secobj_id

Specifies that the audit logs of the specified ID of the secondary object associated with the object needs to be archived.

-secobj_name

Specifies that the audit logs of the specified name of the secondary object associated with the object needs to be archived.

-secobj_revid

Specifies that the audit logs of the specified revision ID of the secondary object associated with the object needs to be archived.

-secobj_seqno

Specifies that the audit logs of the specified sequence number of the secondary object associated with the object needs to be archived.

-error_code

Specifies that the audit logs of the specified error code associated with the failed actions on the object needs to be archived.

-proj_id

Specifies that the audit logs of the specified ID of the project needs to be archived.

-proj_name

Specifies that the audit logs of the specified project name needs to be archived.

-overwrite

Archives permanent audit records (those with days kept value of **-1**). When used in conjunction with the **-delete_record** argument, the permanent audit records are removed from the database.

-h

Displays help for this utility.

ENVIRONMENT

As specified in the *Setting up a Teamcenter command-line environment*.

FILES

As specified in the *Log files produced by Teamcenter*.

RESTRICTIONS

Requires Teamcenter administrator privileges.

EXAMPLES

- To archive audit log entries associated with the **ITAR_license_01** ITAR license for actions performed by the **user1** user belonging to **ADASiteAdministrator** group and created before 30th June 2009:

```
audit_archive -u=Tc-admin-user -p=password -g=group -class=ADA_License
  -type=ITAR_License -id=ITAR_license_01 -owner=user1
  -group=ADASiteAdministrator
  -created_before=30-jun-2009 12:00
```

- To archive audit log entries related to all **__Attach_License** events associated with ITAR licenses attached to the **Part12** item, revision **A**, sequence **2**, created after 31st January 2009:

```
audit_archive -u=Tc-admin-user -p=password -g=group -class=ADA_License
  -type=ITAR_License -event=__Attach_License -secobj_id=Part12
  -secobj_revid=A -secobj_seqno=2
  -created_after=31-jan-2009 12:00
```

- To archive audit log entries associated with **project1** project name for actions performed by **user1** user belonging to **ProjectAdministrator** group and created before 30th September 2010:

```
audit_archive -u=Tc-admin-user -p=password -g=group -class=Requirement
  -type=Requirement -proj_name=pcombine_audit_filesproject1
  -owner=user1 -group= ProjectAdministrator
  -created_before=30-sep-2010 12:00
```

combine_audit_files

Note:

This utility is deprecated and will be removed in a future version.

Combines all the log files into an **TcAuditLog.txt** or **TcAuditLog.xml** file.

SYNTAX

combine_audit_files.pl

ARGUMENTS

source_dir

Specifies the source directory containing audit log files generated during Teamcenter sessions.

target_dir

Specifies the target directory containing the combined audit log file. The target directory must not be the same as the source directory, because the program tries to move the audit files from the source directory to the target directory, combine them, and delete them. The **source_dir** and **target_dir** values can be either an absolute path or a relative path.

ENVIRONMENT

This utility works on any Linux or Windows platforms that install Perl, and the program is in their path.

FILES

The audit log files to be combined (**tc_auditlog_****.txt**, **tc_auditlog_****.xml**) are at **source_dir** directory. The combined master log files (**TcAuditLog.txt**, **TcAuditLog.xml**) are at **target_dir** directory. If the master files **TcAuditLog.txt**, **TcAuditLog.xml** are not found when running this utility, first create them, then append the original audit files to them.

RESTRICTIONS

None.

EXAMPLES

- Suppose the utility **combine_audit_files.pl** and all original audit files are in the current directory, and the master audit files are in the **C:\temp\audit** directory. The program searches all **tc_auditlog_****.xml** and **tc_auditlog_****.txt** files, moves them to the **C:\temp\audit** directory, combines them, and appends them to the **TcAuditLog.xml** and **TcAuditLog.txt** files in the **C:\temp\audit** directory. Finally, all the original audit files that are moved and appended are deleted.

```
perl combine_audit_files.pl . C:\temp\audit
```

- Here, source and target are subdirectories of the current directory. Suppose the utility program is in the current directory, original audit files are in the source directory, and the master audit files are in the target directory. It will look for all **tc_auditlog_****.txt** and **tc_auditlog_****.xml** files in the source directory, move them to the target directory, combine them, and append them to the **TcAuditLog.txt** and **TcAuditLog.xml** files in the target directory. Finally, all original audit files that are moved and appended are deleted.

In general, all audit files are generated by the program and you should not manually edit them. However, in the following situations, you must manually modify the master files using any text editor, such as **vi** or Notepad:

```
perl combine_audit_files.pl source target
```

- If the **TC_audit_delimiter** preference is changed to a value other than the default value (^), you must manually edit the first line of the **TcAuditLog.txt** master audit file to reflect the new delimiter.

For example, if the new delimiter is a dollar sign (\$), the first line of the **TcAuditLog.txt** file must be changed to:

```
ObjectUID$objectId$objectName$revision$objectTypeName$
eventTypeName$userId$loggedDate$properties
```

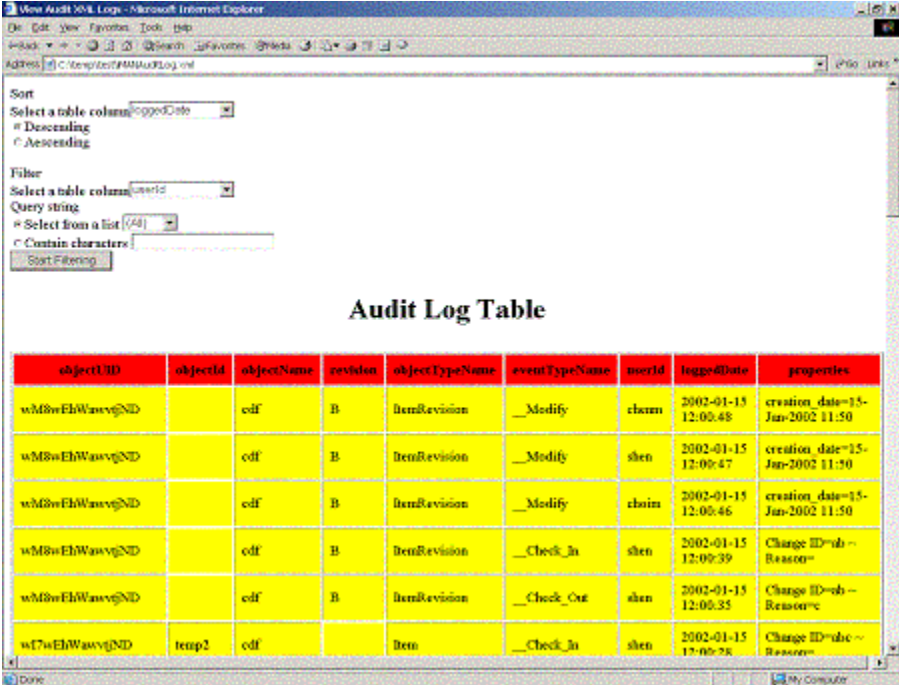
- If the **TC_XML_ENCODING** environment variable in the **tc_profilevars.bat** file is changed to a value other than the default value **iso-8859-1**, you must edit the first line of the **TcAuditLog.xml** master audit file and the **TcAuditLog.xsl** and **TcAuditLogSchema.xsd** files in the sample/audit directory.

For example, if the new encoding is **Shift_JIS** (Japanese), the first line of the **TcAuditLog.xml** file is changed to:

```
<?xml version="1.0" encoding=" Shift_JIS"?>
```

Siemens Digital Industries Software recommends that you run this utility and archive the master audit file periodically, daily, weekly, or monthly, depending upon the data growth. If the master file grows too big it would be difficult to open.

A sample of the XML program files is provided in the **sample/audit/** directory to view the XML audit data in the web browser. The four files (**TcAuditLog.xml**, **TcAuditLog.xsl**, **TcAuditLogSchema.xml**, and **TcAuditLog.js**) must be in the same directory. Opening **TcAuditLog.xml** in Microsoft Internet Explorer 5.5 or higher presents audit data in a table similar to the following that can be sorted and filtered by column.



The screenshot shows a web browser window titled 'View Audit XML Logs - Microsoft Internet Explorer'. The address bar shows 'C:\temp\TcAuditLog.xml'. The page contains a 'Sort' section with a dropdown for 'loggedDate' and radio buttons for 'Descending' (selected) and 'Ascending'. Below is a 'Filter' section with a dropdown for 'userid', a 'Query string' field, a 'Select from a list' dropdown with 'All' selected, and a 'Contain characters' field. A 'Start Filtering' button is at the bottom left of the filter section. The main content is titled 'Audit Log Table' and contains a table with 9 columns: objectID, objectId, objectName, revision, objectTypeName, eventTypeName, userID, loggedDate, and properties. The table has 6 rows of data.

objectID	objectId	objectName	revision	objectTypeName	eventTypeName	userID	loggedDate	properties
wM8wEhWavv@ND		edf	B	ItemRevision	__Modify	rbmm	2002-01-15 12:00:48	creation_date=15-Jan-2002 11:50
wM8wEhWavv@ND		edf	B	ItemRevision	__Modify	shen	2002-01-15 12:00:47	creation_date=15-Jan-2002 11:50
wM8wEhWavv@ND		edf	B	ItemRevision	__Modify	rbmm	2002-01-15 12:00:46	creation_date=15-Jan-2002 11:50
wM8wEhWavv@ND		edf	B	ItemRevision	__Check_In	shen	2002-01-15 12:00:39	Change ID=rb ~ Reason=
wM8wEhWavv@ND		edf	B	ItemRevision	__Check_Out	shen	2002-01-15 12:00:35	Change ID=rb ~ Reason=
wM8wEhWavv@ND	temp2	edf		Item	__Check_In	shen	2002-01-15 12:00:28	Change ID=rb ~ Reason=

Audit data table

The files are described as follows:

TcAuditLog.xml

XML data source of audit records. The file is produced by executing the **combine_audit_files.pl** script.

TcAuditLog.xsl

XML style sheet for displaying the **TcAuditLog.xml** file in the Microsoft Internet Explorer Web browser.

TcAuditLogSchema.xsd

XML schema for defining XML data structure and data types.

TcAuditLog.js

JavaScript for adding dynamic effects to the HTML presentation.

If you use the **TcAuditLog.xml** for purposes other than displaying the audit log on a Web browser, you can modify any of the files. For example, you can modify the **TcAuditLog.xml** file so that the style sheet or schema is not loaded.

None of the files introduced in this section (**combine_audit_files.pl**, **TcAuditLog.xml**, **TcAuditLog.xsl**, **TcAuditLogSchema.xsd**, and **TcAuditLog.js**) require a Teamcenter environment. You can run them anywhere as long as you have Perl and Microsoft Internet Explorer.

define_auditdefs

Note:

This utility and the **install_event_types** utility are deprecated and will be removed in a future version.

Use the Business Modeler IDE to create custom event types. For more information, see *BMIDE for Data Model Design*.

Creates **AuditDefinition** objects in the Teamcenter database. It scans the input file given by the argument **-f** argument.

INPUT FILE FORMAT

The input file contains records to define each AuditDefinition object in the database. Each record is separated by a blank line and conforms to the following format:

```
TYPE_NAME=object-type-name
CLASS_NAME=object-class-name
EVENT_TYPE=event-type-name
PROP_COUNT=property-count /* number of PROP_NAME entries */
PROP_NAME=property-list /* Optional entry */
PROP_NAME=property-list /* Optional entry */
MAX_DAY=max-days-kept
MEDIA_NAME=media-name /* Optional entry */
HANDLER_ID=handler-id /* Optional entry */
```

Tip:

You can see the list of available events in the **Event Type** editor of Business Modeler IDE.

For more information, see *BMIDE for Data Model Design*.

SYNTAX

```
define_auditdefs [-u=user-id {-p=password | -pf=password-file} -g=group]
[-v] [-f=input-file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with **DBA** privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the name of the text file containing the audit definition records.

-v

Specifies verbose mode.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

pom_audit_manager

Allows a site to configure a list of users for whom failed authentication attempts must be logged and provides the ability to later extract that information for analysis.

SYNTAX

```
pom_audit_manager [-u=user-id {-p=password | -pf=password-file} -g=group]
[-install ]
[-report [-before]]
[-purge [-before]]
[-auditable]
[-set event-name user-name ON|OFF ]
[-delete event-name user-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-install

Installs audit functionality.

-report [-before]

Shows audit reports. Optionally, you can use the **-before** argument to show only those records before the specified date.

The **-before** argument must be followed by an 8 digits ISO date such as 20071225 for Christmas day 2007.

-purge [-before]

Permanently removes event records from the user access logging list. Optionally, you can use the **-before** argument to filter purging to only those records before the specified date.

The **-before** argument must be followed by an 8 digits ISO date such as 20071225 for Christmas day 2007.

-auditable

Shows which events are auditable.

Currently **POM_AUDIT_purge_audit**, **POM_AUDIT_bad_password_login**, **POM_AUDIT_bad_password_check**, **POM_AUDIT_change_password** events are auditable.

-set event-name user-name ON|OFF

Adds or removes audit functionality on an event.

-delete event-name user-name

Deletes audit logs for specific events and users.

You can specify the **ALL** value for either event-name or user-name.

Currently **POM_AUDIT_purge_audit**, **POM_AUDIT_bad_password_login**, **POM_AUDIT_bad_password_check**, **POM_AUDIT_change_password** events are auditable.

-h

Displays help for this utility.

ENVIRONMENT

Standard runtime environment only.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

- To see audit reports before a particular date:

```
pom_audit_manager -u=admin -p=admin -g=dba  
-report -before 20071201
```

- To add audit functionality on an event:

```
pom_audit_manager -u=admin -p=admin -g=dba  
-set POM_AUDIT_bad_password_login ON
```

Backup and Recovery

backup_modes

Sets backup mode (read-only, blobby volume, or normal) for the Teamcenter database and volumes to support hot backup by third-party backup systems. Hot backup enables Teamcenter to run in a near-continuous mode even during routine backups.

SYNTAX

```
backup_modes [-u=user-id {-p=password | -pf=password-file} -g=group]
[-pf=password-file]
[-m= {rdonly | normal | blobby | current}] [-f= opencnt] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-m

Sets Teamcenter volumes to read-only, normal, or blobby mode. This argument can also be used to obtain the current backup mode.

rdonly	Places Teamcenter into read-only state. This state holds writing files to the volume during backup.
normal	Places Teamcenter back in normal mode from read-only or blobby volume mode.
blobby	Places Teamcenter in blobby (temporary) volume mode. Teamcenter can be switched into this mode after the third-party backup software takes a snapshot of the data. This allows continuous Teamcenter availability.
current	Returns the current Teamcenter mode.

-f

Obtains information about the Teamcenter volume files opened for the write operation.

-h

Displays help for this utility.

ENVIRONMENT

The proper values must be set for the following preferences:

- **blobbyVolume_NT**
- **blobbyVolume_UNX**
- **TC_enable_backup_modes**

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- Before setting Teamcenter to blobby volume mode, ensure that Teamcenter is in read-only mode.
- You must assign values to the **blobbyVolume_NT** and **blobbyVolume_UNX** preferences even if you are not operating in a heterogeneous environment.

EXAMPLES

- To place Teamcenter volumes in read-only mode, enter the following command:

```
backup_modes -u=Tc-admin-user -p=password -g=group -m=rdonly
```

- To place Teamcenter volumes in normal mode, enter the following command:

```
backup_modes -u=Tc-admin-user -p=password -g=group -m=normal
```

- To place Teamcenter volumes in blobby volume mode, enter the following command:

```
backup_modes -u=Tc-admin-user -p=password -g=group -m=blobby
```

- To obtain the current Teamcenter backup mode, enter the following command:

```
backup_modes -u=Tc-admin-user -p=password -g=group -m=current
```

- To obtain information about Teamcenter volume files opened for write, enter the following command:

```
backup_modes -u=Tc-admin-user -p=password -g=group -f=opencnt
```

backup_xmlinfo

Provides information about Teamcenter volumes defined for a site in XML format. Third-party backup systems require this information for 24x7 hot backup of Teamcenter volumes and databases. The program creates two output files, **backup.xml** and **backup.dtd**, in the directory from which the utility is executed.

SYNTAX

backup_xmlinfo [-u=*user-id* {-p=*password* | -pf=*password-file* -g=*group*} -h

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

Generate backup information for Teamcenter volumes by executing the following command:

```
backup_xmlinfo -u=Tc-admin-user -p=password -g=group
```

The following example illustrates a sample XML output file:

```
<?xml version="1.0" standalone="yes" ?>
<!-- Backup Info : XML File -->
<!DOCTYPE backupInfo SYSTEM "backup.dtd">
<backupInfo>
  <volumeinfo>
    <VolumeName>tokra_vol</VolumeName>
    <VolumeUid>036440ca0b1c558e9f42</VolumeUid>
    <NodeName>ustrw1sun002</NodeName>
    <UnixPath>/netap/tceapps/TCe/TCevols/tokra_vol</UnixPath>
  </volumeinfo>
  <volumeInfo>
    <volumeName>cloudvol1</volumeName>
    <volumeUid>049e5c362cf400002711</volumeUid>
    <nodeName>cloudnode</nodeName>
    <storageType>dss</storageType>
    <externalStorageId>dss-819b89e9b6c3434b9babd34063763306</externalStorageId>
    <cloudPath>cloud</cloudPath>
  </volumeInfo>
  <volumeinfo>
    <VolumeName>satish1_vol</VolumeName>
    <VolumeUid>037840d6b8ac558e9f42</VolumeUid>
    <NodeName>uslvw1097a011</NodeName>
    <WntPath>c:\satish1_vol</WntPath>
  </volumeinfo>
</backupInfo>
```

sfr_instances

Creates and deletes single file recovery instances.

SYNTAX

```
sfr_instances [-u=user-id {-p=password | -pf=password-file} -g=group]
[-d=datasettype] [-ou=owning-user] [-og=owning-group] [-v=volume-name [-ib=any-previous-
backupLabel] -b=new-backup-label [-f=function] {create | delete | list} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-d

Specifies the dataset type to which the specified function applies.

-ou

Specifies the owning user of the datasets to which the specified function applies.

-og

Specifies the owning group of the datasets to which the specified function applies.

-v

Specifies the volume to which the specified function applies.

-ib

Specifies the previous backup label.

-b

Specifies the backup label associated with the create, list or delete function. Use **-b=ALL** to delete all single file recovery instances.

-f= *function*

Specifies the function for the utility. **create** creates the single file recovery instance. **delete** deletes the single file recovery instances.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*. In case this is not already initialized, set the proper values for the following variables to enable further recovery:

TC_sfr_recovery_interval

TC_sfr_process_life_time

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None, however it is a good practice to run this utility after putting Teamcenter in read-only mode and before putting Oracle in Hot backup mode.

EXAMPLES

- To create single file recovery instances associated with the **backup_1** backup label, enter the following command:

```
sfr_instances -u=Tc-admin-user -p=password -g=group -b=backup_1 -f=create
```


- To delete single file recovery instances associated with the **backup_2** backup label, enter the following command:

```
sfr_instances -u=Tc-admin-user -p=password -g=group -b=backup_2 -f=delete
```

- To delete all single file recovery instances in the database, enter the following command:

```
sfr_instances -u=Tc-admin-user -p=password -g=group -b=ALL -f=delete
```

Dispatcher

dispatcher_create_rqst

Provides the ability to create a dispatcher request using command line arguments.

Note:

This utility is a sample utility to be used with datasets. It limits the number of translation arguments to 3. If you need more functionality, you can write a custom ITK using the sample code located in *DISP_ROOT\DispatcherClient\sample\utilities\dispatcher_create_rqst_itk_main.cxx*.

SYNTAX

The syntax of the **dispatcher_create_rqst** utility has two forms:

```
dispatcher_create_rqst
-i=item-ID
-r=revision-ID
[-rn=relation]
-dn=dataset-name
[-dv=dataset-version-number]
-dt=dataset-type-name
-pr= 1 | 2 | 3]
-pn=translator-provider-name
-tn=service-name
-ty=type-string
[-ta1=translation-argument1
 [-ta2=translation-argument2
 [-ta3=translation-argument3]]]
[-u=user-id -p=password | -pf=password-file -g=group]
-verbose | -debug]
```

OR

```
dispatcher_create_rqst
-f=path-name
-dt=dataset-type-name
-pr= 1 | 2 | 3]
-pn=service-provider-name
-tn=service-name
-ty=type-string
[-u=user-id -p=password | -pf=password-file -g=group]
-verbose | -debug]
```

ARGUMENTS

Note:

If a relation is not specified, the **IMAN_specification** relation is used.

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-i

Specifies the item.

-r

Specifies the item revision.

-rn

Specifies the relation name to be used to find the dataset for the given item revision. This argument is optional. If the relation is not specified, the value of **IMAN_specification** is used.

-dn

Specifies the dataset name. This argument is optional.

-dv

Specifies the dataset version number. This argument is optional. If no version number is specified, the specific version that is the latest is used. If a version number is specified, and the version is the latest, only the latest version is used.

Consider the following example. A dataset has the following versions:

Dataset with version number	Description
006122/000	This is the latest version and also the same as version 2.
006122/000:1	This is version one.
006122/000:2	This is version two and the latest specific version.

Consider the following values of the **-dv** argument:

- **-dv=1**: The version 006122/000:1 is used.
- **-dv=2**: The version 006122/000 is used because when you specify the latest version in the **-dv** argument the latest version is used.
- No value specified: The version 006122/000:2 is used because when you do not specify a version in the **-dv** argument, the latest specific version is used.

-dt

Specifies the type of the dataset to be translated

-pr

Specifies the translation priority. Accepted values are **1**, **2**, or **3** corresponding to low, medium and high translation scheduler priority.

-pn

Specifies the name of the translator provider, for example, Siemens.

-tn

Specifies the name of the translator service, for example **ideastojt**.

-ty

Specifies the type name, for example **COMMANDLINE**.

-verbose

Provides additional information. This argument is optional.

-f

Specifies an input file used to create one or more dispatcher requests. This argument is used in lieu of the item, revision, relation name, dataset name, and dataset version arguments. This argument is optional.

The format for the input file is as follows:

```
<item ID>,<revision ID>,[relation name],[dataset name],  
[version number]
```

Note:

Commas are required.

-ta1

Specifies a translation argument.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To create a dispatcher request for the latest version of the I-deas part dataset related to the **Block/A** item revision, enter the following command on a single line:

```
dispatcher_create_rqst -u=user_name -p=user_name -g=dba  
-i=Block -r=A -dt=IdeasPart -pr=2 -pn=Siemens  
-tn=ideastojt -ty=COMMAND_LINE
```

- To create a dispatcher request for version 2 of the **Block/A** item revision, enter the following command on a single line:

```
dispatcher_create_rqst -u=user_name -p=user_name -g=dba  
-i=Block -r=A -dv=2 -dt=IdeasPart -pr=2 -pn=Siemens  
-tn=ideastojt -ty=COMMAND_LINE
```

- To create two Siemens Digital Industries Software **ideastojt** dispatcher requests with a priority of 2:
 - One request for the latest version of the **Block** dataset, an **IdeasPart** or **IdeasAssembly** type dataset, associated with item revision **Block/A** by an **IMAN_specification** relation;
 - One for the latest version of the **Asm** dataset, an **IdeasPart** or **IdeasAssembly** type dataset associated with item revision **Asm/A** by an **IMAN_specification** relation, enter the following command on a single line:

```
dispatcher_create_rqst -u=user_name -p=user_name -g=dba
-f=ctrl -dt=IdeasPart,IdeasAssembly -pr=2 -pn-Siemens
-tn=ideastojt -ty=COMMAND_LINE
```

The lines in the input file are:

```
Block,A,,Block,
Asm,A,,Asm,
```

Note:

All parts in the file are subject to the same translation, because the translator is specified on the command line.

dispatcher_util

Allows you to list, delete or resubmit dispatcher requests.

You must have permissions to manage dispatcher requests. For more information, see *Create a dispatcher client access rule*.

The *Dispatcher-client-proxy-user* used by Dispatcher Client has all the permissions required to manage dispatcher requests.

SYNTAX

dispatcher_util

```
[-u=user-id {-p=password | -pf=password-file} -g=group]
-a=list | delete | resubmit
[-force]
[-export=file:file path]
[-taskid=dispatcher task ID | file:file path]
[-provider=provider name]
[-service=service name]
[-priority=priority number]
[-state=dispatcher state]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-a

Performs the list, delete or resubmit actions.

Choose from the following action values:

- **list**: Lists dispatcher requests.
- **delete**: Deletes dispatcher requests.
- **resubmit**: Resubmits dispatcher requests.

-force

Forces an action without any prompts.

-export

Exports the information in a file.

-taskid

Specifies the task ID of the dispatcher request.

-provider

Specifies the name of the dispatcher provider, for example, Siemens.

-service

Specifies the service name, for example, tozipfile.

-priority

Specifies the dispatcher priority. Accepted values are **1**, **2**, or **3** corresponding to low, medium and high priority.

-state

Specifies the dispatcher state.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- List all high priority dispatcher requests.

```
dispatcher_util -u=Dispatcher-client-proxy-user -p=password -a=list -priority=3
```

- Delete all dispatcher requests in the **INITIAL** state.

```
dispatcher_util -u=Dispatcher-client-proxy-user -p=password -a=delete -state=INITIAL
```

- Resubmit all **tozipfile** dispatcher requests.

```
dispatcher_util -u=Dispatcher-client-proxy-user -p=password -a=resubmit -service=tozipfile
```

Migration

convert_distribution_lists

Converts distribution lists to alias lists. Also allows users to create an alias list by importing data from a text file.

SYNTAX

```
convert_distribution_lists [-u=user-id {-p=password | -pf=password-file} -g=group]
[-all] [-delete] [-dist_list_name=distribution-list-name]
[-import_file=file-name] [-import=file-name] [-new_list_name=alias-list-name] [-overwrite] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-all

Converts all the distribution lists to alias lists.

-delete

Deletes distribution lists that were converted to alias lists.

-dist_list_name

Converts a specified distribution list to an alias list.

-import

Creates an alias list from the addresses specified in the file. The format of the ASCII file is:

```
abc1@xyz.com
abc2@xyz.com
abc3@xyz.com
```

-new_list_name

Specifies the name of the new alias list. This argument must be used with the **-import** option.

-overwrite

Overwrites the existing alias list.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To convert all distribution lists in the database to alias lists, enter the following command on a single line:

```
convert_distribution_lists -all
```

- To create an alias list using the distribution list **marketing_list** and then delete the distribution list:

```
convert_distribution_lists -dist_list_name=marketing_list -delete
```

- To create a new alias list with the name **Local_Alias_List**, populate the list with the addresses listed in the **address_local.txt** file, and overwrite the existing address list, enter the following command on a single line:

```
convert_distribution_lists -import=address_local.txt  
-new_file_name=Local_Alias_List -overwrite
```

move_mso_forms

Finds all forms of type **OfficeDocForm** that are directly attached to folders, items, or item revisions and moves them to a corresponding dataset as a named reference when upgrading Teamcenter Engineering 8.x and 9.x databases to a Teamcenter database. These forms are used for property synchronization between Microsoft Office and the Teamcenter database.

Note:

This utility is called by the upgrade script when upgrading from a previous version of Teamcenter Engineering to Teamcenter.

SYNTAX

```
move_mso_forms [-u=user-id {-p=password | -pf=password-file} -g=group] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

migrate_vendor_part_locations

Migrates the locations of vendor parts from the **Provided From Relations** property to the **Typed Reference** property. You can specify whether to migrate only the first location by using the **migratefirst** argument, or make copies of vendor parts each time the utility finds multiple locations by using the **migrateall** argument.

SYNTAX

```
migrate_vendor_part_locations [-u=username {-p=password | -pf=password-file} -g=groupname]
[-option=[list|migratefirst|migrateall]] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option

Use the **list** value to find applicable vendor parts and print their information in the log file. Use the **migratefirst** value to migrate only the first location of the applicable vendor parts. Use the

migrateall value to make copies of vendor parts each time the utility finds multiple locations in the applicable vendor parts.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

ums_mapping

As of version 2406, Teamcenter employs an updated unit management system (UMS). Following an upgrade of a Teamcenter installation from a version prior to 2406, use this utility to map legacy BMIDE units of measure to definitions in the UMS.

SYNTAX

```
ums_mapping [-u=user-id {-p=password | -pf=password-file} -g=group]
[-analyze {-report_file= | -input_file=}config_file] |
[-map -input_file=config_file -report_file=map_report_file] |
[-unmap {-input_file=config_file | -all} -report_file=unmap_report_file]
[-create_uom -report_file=create_uom_report_file]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

To use the **-map**, **-unmap**, and **-create_uom** arguments, this is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

To use the **-map**, **-unmap**, and **-create_uom** arguments, this is a user who is a member of a group with DBA privileges.

If used without a value, the user's default group is assumed.

-analyze

Evaluates the legacy existing units of measurement system.

Use **-analyze** with **-report_file** to examine the Teamcenter database for existing units of measure and their definitions. The utility attempts to map legacy units of measure to units of measure in the UMS. This mapping is performed based on unit names and symbols.

The utility generates the .csv file specified by **-report_file=config_file** showing these mappings along with units that could not be mapped between the systems. Confirm and update the mappings in this file as described in Migrate legacy units of measure.

The .csv file contains the following mapping information:

Column	Definition
Is Approved	Remove the # character in this column to process this unit when using -map , -unMap , or -createMap .
UOM Name	The name of the unit in the legacy system.
UOM Symbol	The symbol of the unit in the legacy system.
In Use	A value of Yes indicates that this unit is used by an object in the legacy system.
Action	The utility action performed to create the file.
Status	The unit's mapping state at the time the file was created.
Details	A description of the unit's state and any recommended actions.
Unit Definition ID	The ID of the unit in the UMS.
Unit Definition Name	The name of the unit in the UMS.
Unit Definition Symbol	The symbol of the unit in the UMS.

After confirming and completing the mappings in the .csv file, use **-analyze** with **-input_file** to generate a dry-run report showing what the results will be if you use the .csv file to map the approved mappings.

-map

Updates the Teamcenter database references between the legacy unit of measure line items with the UMS line items defined in the configuration file specified with **-input_file**.

Approved units in the file (units with no "#" symbol in the first column) are updated in the Teamcenter database. The utility generates the report specified with **-report_file** listing the updates and their completion status.

-unmap

Removes Teamcenter database relationships between legacy units of measure and the UMS.

Using **-input_file** with **-unmap** removes specific unit of measure relationships. Relationships for approved units in the file (units with no "#" symbol in the first column) are removed from the Teamcenter database.

Using **-all** with **-unmap** removes all existing units of measure relationships from the Teamcenter database. You could then restart migrating legacy units of measure as described in Migrate legacy units of measure to the unit management system.

The results of either removal is detailed in the file specified by **-report_file**.

-create_uom

Analyzes the UMS and creates new units for definitions without related units. Results are listed in the report specified by **-report_file**.

-h

Displays help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment.

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

None.

EXAMPLES

- Analyze the Teamcenter database for existing UMS unit definitions. Create a mapping configuration file containing mappings between the legacy units of measure to existing unit of measure definitions in the UMS.

```
ums_mapping -u=username -p=password -analyze
-report_file=c:\temp\mapping_config.csv
```

- Perform a dry run of linking approved units of measure to unit definitions in the UMS.

```
ums_mapping -u=username -p=password -analyze
-report_file=c:\temp\mapping_config.csv
```

- Add unit of measure relationships to the Teamcenter database.

```
ums_mapping -u=admin-username -p=admin-password
-g=dba -map -input_file=c:\temp\mapping_config.csv
-report_file=c:\temp\mapping_report.csv
```

The utility generates the report *mapping_report.csv* listing the updates and their completion status.

- Remove unit of measure relationships from the Teamcenter database

```
ums_mapping -u=admin-username -p=admin-password
-g=dba -unmap -input_file=c:\temp\mapping_config.csv
-report_file=c:\temp\mapping_report.csv
```

The utility generates the report *mapping_report.csv* listing the removed unit relationships.

- Create new units of measure and add the new units to the Teamcenter database.

```
ums_mapping -u=admin-username -p=admin-password -g=dba -create_uom
-report_file=c:\temp\unit_report.csv
```

The utility generates the report *unit_report.csv* listing the created units.

Portfolio, Program, and Project Management

create_project

Creates or updates projects or programs in the database based on command line input or input from a text file. For example, you can use this utility to:

- Create a project using either a command line (shown below) or an input file.

```
create_project [-u=user-id {-p=password | -pf=password-file} -g=group]
[{-id=project-id -name=project-name
[-desc=project-description -status=A | I]
-teams=group1~role1~user1~group2~role2~user2...
[-privileged=group1~role1~user1~group2~role2~user2... ]
[-teamadmin=group~role~user]} [-projectcategory=project-category]
```

- Update a project using either a command line (shown below) or an input file.

```
create_project [-u=user-id {-p=password | -pf=password-file} -g=group]
{-update {-updateproject=project-id[-id=new-id][ -name=new-name]
[-desc=new-description][ -status=A | I]
[-addmember=group~role~user][ -removemember=group~role~user]
[-setprivuser=group~role~user][ -unsetprivuser=group~role~user]
[-setteamadmin=group~role~user]
[-unsetteamadmin=group~role~user]
[-changeprojectadmin=user1][ -defaultprojforuser=group~role~user]}
[-projectcategory= project-category] [-collabcategory= comma-delimited-string-of-
collaboration-categories]
```

- **Create a program** using either a command line (shown below) or an input file.

```
create_project [-u=user-id {-p=password | -pf=password-file} -g=group]
[{-id=project-id -name=project-name [-desc=project-description -status=A | I]
-teams=group1~role1~user1~group2~role2~user2...
[-privileged=group1~role1~user1~group2~role2~user2... ]
[-teamadmin=group~role~user]} [-projectcategory=project-category]
[-program=y]
```

- **Update a project to a program** using either a command line (shown below) or an input file.

```
create_project [-u=user-id {-p=password | -pf=password-file} -g=group]
-update -updateproject=project-id -program=y
```

SYNTAX

```
create_project [-u=user-id {-p=password | -pf=password-file} -g=group]
arguments [-h
```

arguments can include one or more of the following:

- **-addmember**=*group~role~user-id*
- **-changeprojectadmin**=*user1*
- **-collabcategory**=*comma-delimited-string-of-collaboration-categories*
- **-defaultprojforuser**=*group~role~user*
- **-delimiter**=*delimiter-character*
- **-desc**=*project-description*
- **-id**=*project-id*
- **-input**=*text-input-file*
- **-name**=*project-name*
- **-privileged**=*group1~role1~user1...*
- **-program**=*y|n*
- **-projectcategory**=*project-category*
- **-removemember**=*group~role~user*
- **-setprivuser**=*group~role~user*
- **-setteamadmin**=*group~role~user*
- **-status**=*A|I*
- **-teamadmin**=*group~role~user*
- **-teams**=*group1~role1~user1~group2~role2~user2...*
- **-unsetprivuser**=*group~role~user*
- **-unsetteamadmin**=*group~role~user*
- **-update**
- **-updateproject**=*project-id*

ARGUMENTS

-u Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf Specifies the **password file**. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-id* value to be the password.

This argument is mutually exclusive with the **-p** argument.

-g Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h Displays help for this utility.

arguments Specifies one or more arguments.

-addmember

Specifies adding a group member object.

-changeprojectadmin

Specifies changing the administrator rights on a project.

-collabcategory

Note:

This field is used by Mechatronics Process Management. Refer to the Mechatronics Process Management online documentation for more information.

Specifies collaboration category.

-defaultprojforuser

Specifies setting the selected project as the default project.

-delimiter

Specifies the delimiting character between fields in the text file, which has to be one character. The default character is |.

Note:

The project and collaboration categories are validated against their respective LOV values, as defined in the **Fnd0ProjectCategories** and **Fnd0CollaborationCategories** LOVs in the Business Modeler IDE.

- Format of the text file assuming default delimiter (|):

```
delimiter- id|name|desc|status|teams|privileged|team-admin|
          project-category|collabcategory1,collabcategory2
```

- Format of the text file with an empty description:

```
description- id|name| |status|teams|privileged|team-admin|
            project-category|comma-delimited-string-of-collaboration-categories>
```

-desc

Specifies the project description.

-id

Specifies the ID of the project.

-input

Specifies a text file containing multiple entries of project id, project name, active (A) or inactive (I) indicator, project description, teams, privileged members, and optional team administrator.

- When using the *create* option to create multiple projects, the syntax of the input file is as follows:

```
create_project -u=Tc-admin-user -p=password -g=group
-input=file_name_4_create
```

The syntax of the **file_name_4_create** file is as follows:

```
# id|name|desc|AorI|teams|priv|teamadmin|projectcategory|
collabcategory|Program
proj_LCS-74838_1|proj_LCS-74838_1|proj_LCS-74838_1|A|
dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner|SI
proj_LCS-74838_2|proj_LCS-74838_2|proj_LCS-74838_2|A|
```



```
dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin| |SI
proj_LCS-74838_3|proj_LCS-74838_3| |A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin| |SI
```

- When using the *update* option to update properties of existing projects, the syntax of the input file is as follows:

```
create_project -u=Tc-admin-user -p=password -g=group -update
-input=file_name_4_update
```

The syntax of the **file_name_4_update** file is as follows:

```
-updateproject=proj_LCS-74838_1|-id=proj_LCS-74838_1|-
name=proj_LCS-74838_1|-desc=updated for proj_LCS-74838_1|-
collabcategory=SI
-updateproject=proj_LCS-74838_2|-id=proj_LCS-74838_2|-
name=proj_LCS-74838_2|-desc=updated for proj_LCS-74838_2|-
projectcategory=Partner
```

- When using this option to add members to projects, the syntax of the input file is as follows:

```
-updateproject=proj_id1>|-addmember=Group~Role~user_id
-updateproject=proj_id2|-addmember=Group~Role~user_id
```

-name

Specifies the name of the project.

-privileged

Defines privileged group members.

-program

Specifies whether a program is being created.

-projectcategory

Specifies project category. Default values are **Internal**, **Partner**, and **Supplier**.

If you want to change the listed project categories, you can change the values in the list of values (LOV). Project categories are defined in the **Fnd0ProjectCategories** LOV found in the Business Modeler IDE.

-removemember

Specifies removing a group member object.

-setprivuser

Specifies setting privileged access to users in a given project.

-setteamadmin

Specifies granting administrator rights to a user in a project.

-status

Specifies the status, either active (**A**) or inactive (**I**).

-teamadmin

Specifies the team administrator.

If not specified, the default team administrator is the logged-on user who is running the utility.

-teams

Specifies group members to be on the project team. This argument accepts valid user, role, and group names. Use the tilde character (~) as a delimiter when specifying group, role, and user, as follows:

```
-teams=group1~role1~user1~group2~role2~user2...
```

In addition, you can specify all members in a group as follows:

```
-teams=group1~*~*
```

-unsetprivuser

Specifies revoking access to users in a given project.

-unsetteamadmin

Specifies revoking administrator rights to users in a given project.

-update

Specifies updates to either a project or a program.

-updateproject

Specifies the specific project for updates.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

The input file must conform to the syntax described in the **-input** argument description.

RESTRICTIONS

None.

RETURN VALUES

Return value upon success 0

Return value upon failure >1

EXAMPLES

Note:

Creating projects does not support building hierarchy.

- To create a single project with ID **123456**, named **ABC Car 123 Model**, description **A high end version of ABC Car**, with an active status assigned to **Car 1** and **Car 2** groups, enter the following command on a single line:

```
create_project -u=Tc-admin-user -p=password -g=group-name -id=123456
-name="ABC Car 123 Model" -desc="A high end version of ABC Car"
-status=A -teams="Car 1~Designer~Smith~"Car2"-Engineer~Jones
```

- To create multiple projects from an input file using the **create_project** utility:
 - Create an input file, **project_input_file.txt**, containing the project data for each project:

```
# id|name|desc|AorI|teams|priv|teamadmin|projectcategory|
collabcategory|Program
proj_LCS-74838_1|proj_LCS-74838_1|proj_LCS-74838_1|A|
dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner|SI
proj_LCS-74838_2|proj_LCS-74838_2|proj_LCS-74838_2|A|
dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin| |SI
proj_LCS-74838_3|proj_LCS-74838_3| |A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin| |SI
```

- Enter the following command on a single line:

```
create_project -u=Tc-admin-user -p=password -g=group-name -input=/tmp/
project_input_file.txt
```

This example creates three projects: **proj_LCS-74838_1**, **proj_LCS-74838_2**, and **proj_LCS-74838_3**.

- To update metadata with the **create_project** utility using an input file:
 - Create an input file, **update_metadata_input_file.txt**, containing the project metadata that you want to update:

```
-updateproject=N001| -id=N001AA| -name=N001AA| -desc=N001AA| -status=A
-updateproject=PSLV07| -id=PSLV07AA| -name=PSLV07AA| -desc=PSLV07AA| -
status=A
```

2. Run the **create_project** command:

```
create_project -u=Tc-admin-user -p=password -g=group -update
-input=C:\apps\tc\project\update_metadata_input_file.txt
```

This example produces the following updated project IDs:

- **N001→N001AA**
 - **PSLV07→PSLV07AA**
- To change the administrator using the **create_project** utility input file option:
 1. Create an input file, **change_project_admin_input_file.txt**, containing the new project administrator data:

```
-updateproject=N001AA| -changeprojectadmin=ed_admin
-updateproject=N021| -changeprojectadmin=ed_admin
```

2. Enter the following command on a single line:

```
create_project -u=Tc-admin-user -p=password -g=group -update
-input=C:\apps\tc\project\change_project_admin_input_file.txt
```

This example changes the project administrator for both the N001AA and N021 projects to **tcadmin**.

- Create a program:

- **Command line**

```
create_project -u=Tc-admin-user -p=password -g=group -id=ProjID
-name=ProjName -program=y
```

- **Input file**

1. Create an input file, **create_program_file.txt**, containing the program information for each project:

```
# id|name|desc|AorI|teams|priv|teamadmin|projectcategory|
collabcategory|Program
```

```

program_1|program_1|program_1|A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner| |y
program_2|program_2|program_2|A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner| |y
program_3|program_3|program_3|A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner| |y
program_4|program_4|program_4|A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner| |y
program_5|program_5|program_5|A|dba~DBA~ed_admin|
dba~DBA~ed_admin|dba~DBA~ed_admin|Partner| |y

```

2. Enter the following command on a single line:

```

create_project -u=Tc-admin-user -p=password -g=group -input=/tmp/
create_program_file.txt

```

- Update a project to a program:

- **Command line**

```

create_project -u=Tc-admin-user -p=password -g=group -update
-updateproject=ProjID -program=y

```

- **Input file**

1. Create an input file, **update_program_file.txt**, containing the program information for each project:

```

-updateproject=project_1| -program=y
-updateproject=project_2| -program=y

```

2. Enter the following command on a single line:

```

create_project -u=Tc-admin-user -p=password -g=group -update
-input=/tmp/update_program_file.txt

```

update_project_data

Updates project data in the Teamcenter database.

Note:

The **-f=update** option initiates the update of all project-related data in a database. This process can take a long time depending on the number of objects assigned to projects. When specifying the project ID (**-project**) for one or more projects, the action applies only the given projects. When updating specific projects, you may need to update other project-related data by running the utility again.

SYNTAX

```
update_project_data [-u=user-id {-p=password | -pf=password_file} -g=group_name
[-f=function] [-force (for update only)]
[-project= project-ID1[,project-ID2...]] [-program= program-ID1[,project-ID2...]]
[-operation_name=
{ "CheckIn","CheckOut","Create","Delete","Export","Import","Revise","Save","SaveAs","All" }]
[-type={any workspace object} {-ID=item-id_or_item-rev-id | -key=key-id}] [-rev-id=revision-id]
[-owning_user=owning_user_id] [-owning_group=owning_group_name] [-name=
name_of_object_in_search] [-file=input_file_with_workspace_object_uids_to_process] -h
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the function performed by the utility. Must be one of the following options:

- **update -force** | **-project=project_ID** | **-project=projectID1[,projectID2...]**

(Default function) Updates project-related data. Use this function to:

- Update project-related data in the database after modifying site propagation rules.
- Clean up corrupted project-related data that may have been caused by system crashes.

-force

Specifies that all projects in the database are unconditionally updated using the current site propagation rules. This process may take a long time for a database with projects containing a large number of objects.

-project=project_ID

Unconditionally updates the specified project in the database using the current site propagation rules.

-project=projectID1[,projectID2...]

Unconditionally updates the specified list of projects in the database using the current site propagation rules.

For example, for the action performed on two projects (**Proj4000** and **Proj5000**):

-project=Proj4000,Proj5000

Use double quotation marks (") if any project ID contains one or more blank spaces.

See the **Examples** section below for examples on using the **-force** function.

- **update_hierarchy**

Updates project-related data according to project hierarchy.

Use this to update project-related data in a database after creating project hierarchy. It updates the owning project and project properties of the data according to the project hierarchy.

Use the **-dry** argument to generate a report, which is useful to see how data looks if the utility is run.

See the **Examples** section below for examples on using the **update_hierarchy** function.

- **delete -project=project ID1[,project ID2...]**

Deletes one or more projects identified by the **-project** argument. Only the Project Administrator of each project that is to be deleted can use this function.

This process may take a long time, depending on the number of objects in each project.

See the **Examples** section below for examples on using the **delete** function.

-project

Specifies the project of the program for which data needs to be updated. When updating the hierarchy, you must specify the program ID.

-program

Specifies the program for which data needs to be updated. When the program security attribute on a project is set to **true**, Teamcenter considers the project as a program, and program-level access rules are applied.

-operation_name

The name of operation must be one of the supported operations:

"CheckIn","CheckOut","Create","Delete","Export","Import","Revise","Save","SaveAs","All".

-type -ID | -key | -rev_id

Indicates the type of the source object, which can be any workspace object.

-ID

Specifies the ID of the source item or item revision.

-key

Indicates the key ID of the object.

-rev_id

Specifies the ID of the source item revision. If omitted, the default is the latest revision.

-name -owning_group | -owning_user

Indicates the name of the source object.

This is required if **-ID** is not supplied.

-owning_group

Specifies the owning group of the source objects where propagation should be started.

-owning_user

Specifies the owning user of the source object where propagation should be started.

-file

Updates propagation-related data in the database for given objects.

The object UIDs must be provided in a separate line in the file. This option should not be used with other options.

See the **Examples** section below for examples on using the **-file** argument.

-h

Displays help for this utility.

MIGRATE TO A SINGLE-LEVEL HIERARCHICAL PROGRAM

1. Select an existing business situation to migrate to a single-level hierarchical program. For example:

```
Program_M1 (Program =True) - D1 (owning_program= Program_M1,
Project IDs= Program_M1)
Program_M2 (Program =True) - D2 (owning_program= Program_M2, Project
IDs= Program_M2)
Program_M3 (Program =True) - D3 (owning_program= Program_M3, Project
IDs= Program_M3)
```

2. Change the program to a project, which you would like to add as child under program to build the hierarchy.

```
Program_M1 (Program =False) - D1 (owning_program= Program_M1,
Project IDs= Program_M1)
Program_M2 (Program =False) - D2 (owning_program= Program_M2,
Project IDs= Program_M2)
Program_M3 (Program =False) - D3 (owning_program= Program_M3,
Project IDs= Program_M3)
```

3. Add the projects from Step 2 under an existing program or new program to build the hierarchy.

Top Program (Program=True)

```
Program_M1 (Program =False) - D1 (owning_program= Top
Program.Program_M1, Project IDs= Program_M1)
Program_M2 (Program =False) - D2 (owning_program= Top
Program.Program_M2, Project IDs= Program_M2)
```

```
Program_M3 (Program =False) - D3 (owning_program= Top
Program.Program_M3, Project IDs= Program_M3)
```

4. Align the data for the newly created hierarchy in Step 3.

Consider the following:

- Run the **update_project_data** utility in the mode appropriate for the volume of data. This utility can process single program or multiple programs or a single project or multiple projects.
- Use the switch that is appropriate based on the volume of data to process. For example, the dry run (**-dry**) switch generates a report on the data state after migration.

The following example illustrates running the **update_project_data** utility for a single project:

```
apps\tc\tc13\TR\tc_menu>update_project_data -u=Tc-admin-user -p=password
-g=group -f=update_hierarchy -project= Program_M1
```

```
C:\apps\tc\tc13\TR\tc_menu>update_project_data -u=Tc-admin-user
-p=password -g=group -f=update_hierarchy -project= Program_M2
```

```
C:\apps\tc\tc13\TR\tc_menu>update_project_data -u=Tc-admin-user
-p=password -g=group -f=update_hierarchy -project= Program_M3
```

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

EXAMPLES

- The following examples illustrate the use of the **-f=update** function:
 - Enter the following command to update the propagation data using the source object of type Item with a specified Item ID:

```
-f=update -type=Item -ID=ItemID
```

- Enter the following command to update the propagation data using the latest item revision of Item with a specified item ID as the source object:

```
-f=update -type=ItemRevision -ID=ItemID
```

- Enter the following command to update the propagation data using item revision with a specified item ID and revision ID as the source object:

```
-f=update -type=ItemRevision -ID=ItemID -rev_id=rev_id
```

- Enter the following command to update the propagation data using item with the specified key as the source object; refer to the [get_key_string utility](#).

```
-f=update -type=Item -key=key-of-item
```

- Enter the following command to unconditionally update the specified project in the database using the current site propagation rules:

```
-f=update -project=project-id
```

- Enter the following command to unconditionally update the specified list of projects using the current site propagation rules:

```
-f=update -project=project-id1[,project-id2...]
```

The project IDs are given as a comma-separated string. For example, **-project="Proj4000,Proj5000"** specifies that the action is performed on two projects: **Proj4000** and **Proj5000**.

- Enter the following command to update all projects in the database using the current site propagation rules:

```
-f=update
```

This command is normally used to update all project data after site propagation rules have been modified. The update algorithm updates project data for objects with a last project assignment date prior to the last site propagation rule modification date.

- Enter the following command to unconditionally update all projects in the database using the current site propagation rules:

```
-f=update -force
```

Note:

If the database contains a large number of projects, processing time could be considerable.

- Enter the following command to align the data specific to the projects under the **Program001** program hierarchy:

```
-f=update_hierarchy -project=Program001
```

- Enter the following command to align the data to program project hierarchy unconditionally for all programs:

`-f=update_hierarchy`

Note:

If the database contains a large number of projects, processing time could be considerable.

- The following example illustrates a SQL query to generate a file of UIDs.

Query to get all the workspace object UIDs that have own **ProjectObjectRelation**.

Note:

This query is the default query used in utility. It runs on all the objects having own **ProjectObjectRelation**. This query is not sufficient to execute only on a set of few objects.

You can add more criteria to the SQL and scope the objects you want to process.

```
select/*+PARALLEL*/ distinct trop_level_proj_obju from
PROJECTOBJECTRELATION;
```

Caution:

You must run this utility on the entire data object. If you only run the utility on few objects, then you should be aware that there is data that will not align with latest propagation rules. Only in exceptional cases where the volume of data is very high and takes a long time to run, you should scope in the objects by generating your own UID file.

- The following are examples of using the `-f=update_hierarchy` function:

- Usage 1. `-f=update_hierarchy -program=program-ID`

When `-program=program-ID` is used with this function, all the data of each of the child projects of `program-ID` are updated as per the newly created hierarchy.

- Usage 2. `-f=update_hierarchy -program=program-ID1, program-ID2...`

When `-program=program-ID1, program-ID2...` is used with this function, all the data of each of the child projects of the specified programs are updated as per the newly created hierarchy.

- Usage 3. `-f=update_hierarchy -project=project-ID`

When `-project=project-ID` is used with this function, all the project data is updated as per the newly created hierarchy.

- Usage 4. **-f=update_hierarchy -project=project-ID1, project-ID2...**

When **-project=** is used with multiple projects with this function, all the data of all the projects specified in the argument is updated as per the newly created hierarchy.

- Usage 5. **-f=update_hierarchy**

When there is no **-project** or **-program** used with this function, all the data in the system is updated as per the newly created hierarchy.

Note:

This is not recommended for a large set of data.

- Usage 6. **-f=update_hierarchy -dry**

When using the **-dry** argument with **update_hierarchy**, a report is generated; however, no data is actually updated.

Object	Existing Owning Project	Existing Projects	Updated Owning Project	Updated Projects	Result	Comments
vega	NULL	testproj1 testproj12		testproj1 testproj12	FAIL	object is not updated
vega1	NULL	testproj1 testproj12		testproj1 testproj12	FAIL	object is not updated
vega3	NULL	testproj1 testproj12		testproj1 testproj12	FAIL	object is not updated

update_propagation_data

Propagates security data from top level objects to other related workspace objects based on current site propagation rules. Execute this utility when a propagation rule is added or an existing propagation rule is removed.

SYNTAX

```
update_propagation_data [-u=user-id {-p=password } -g=group] [-file=input file path]
[-outputUIDfile=output file path] -h
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Execute the following command if utility is not in the Authorization application.

```
install_authorization_rules -u=Tc-admin-user -p=password -g=group -install
```

The script will add update_propagation_data utility in Authorization application and system administrator/dba can provide full access to certain group/role.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the user ID value to be the password.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Path of the input file containing UIDs of top level objects having own **ProjectObjectRelation**(POR).

Use SQL queries to get the list of object uids to process. Please refer **Examples** section.

Utility propagates security data from these top level objects to other related workspace objects based on current site propagation rules.

The file should not contain more than 50K UIDs. Split the file in multiple files each having 50K UIDs maximum.

The object UIDs must be provided in a separate line in the file. This option should not be used with other options.

-outputUIDfile

Path of the output file which will be populated with the UIDs of top level objects having own **ProjectObjectRelation**. This file can be used as input value for **-file** option. This option should not be used with other options.

Note:

You can use **-outputUIDfile** option, which will generate the UIDs of all top level objects having own **ProjectObjectRelation** or you can create your own UID file using SQL query.

-h

Displays help for this utility.

EXAMPLES

The following example illustrates a SQL query to generate a file of UIDs.

Query to get all the workspace object UIDs that have own **ProjectObjectRelation**.

Note:

This query is the default query used in utility. It will run on all the objects having own **ProjectObjectRelation**. This query is not sufficient to execute only on a set of few objects.

You can add more criteria to the SQL and scope the objects you want to process.

```
select/*+PARALLEL*/distinct RTOP_LEVEL_PROJ_OBJU from
PROJECTOBJECTRELATION;
```

Caution:

Utility must be run on the entire data object. If you only run the utility on few objects, then you should be aware that there is data that will not align with latest propagation rules. Only in exceptional cases where the volume of data is very high and takes a long time to run, you should scope in the objects by generating your own UID file.

Server manager

installmgr.bat

Installs the server manager for four-tier environments as a Windows service. A copy of this script is created in the directory `TC_ROOT\pool_manager\confs\configuration-name` by Teamcenter Environment Manager (TEM) for each installed server manager instance. Replace *configuration-name* with the server manager configuration name. The name consists of the TEM configuration name and the server manager's pool ID. This script is run automatically by TEM but can be run manually if required.

SYNTAX

installmgr

ARGUMENTS

None.

ENVIRONMENT

None.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

To manually install a server manager named **PoolA** in a Teamcenter configuration named **MyConfig**, enter the following command on a single line:

```
D:\tc_root\pool_manager\confs\MyConfig_PoolA>installmgr
```

uninstallmgr.bat

Uninstalls the server manager for four-tier environments as a Windows service. A copy of this script is created in the directory `TC_ROOT\pool_manager\confs\configuration-name` by Teamcenter Environment Manager (TEM) for each installed server manager instance. Replace *configuration-name* with the server manager configuration name. The name consists of the TEM configuration name and the server manager's pool ID.

SYNTAX

uninstallmgr

ARGUMENTS

None.

ENVIRONMENT

None.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

To manually uninstall a server manager named **PoolA** in a Teamcenter configuration named **MyConfig**, enter the following command on a single line:

```
D:\tc_root\pool_manager\confs\MyConfig_PoolA>uninstallmgr
```

mgrstop

Shell script to cleanly stop the server manager for J2EE four-tier environments. A copy of this script is created in the directory `TC_ROOT/pool_manager/confs/configuration-name` by Teamcenter Environment Manager (TEM) for each installed server manager instance. Replace *configuration-name* with the server manager name. The name consists of the TEM configuration name and the server manager's pool ID. On Linux systems, this script is launched by `rc.tc.mgr_config` when the stop action is requested.

Note:

This command stops a server manager instance; it does not mark a Windows service as stopped. If the server manager is running as a Windows service, the Windows Services Manager attempts to restart the manager if the service is configured for restarting. A server manager running as a Windows service must be stopped from the Windows Services Manager.

SYNTAX

mgrstop [-Dimmediate]

ARGUMENTS

-Dimmediate

Specifies whether the manager terminates all servers immediately rather than waiting for them to become idle (which is the default behavior).

ENVIRONMENT

None.

FILES

Files in the `TC_ROOT/pool_manager/confs/config` directory.

File	Description
mgr.tmp	Temporary file written by the manager at startup and deleted when it exits. The file contains the values of the POOL_ID and the JMX_HTTP_ADAPTOR_PORT environment variables, which identify the server manager pool and the port on which it listens.
password.txt (optional)	Contains the user name and the encrypted password for the server manager administration interface. This file exists only if the user name or password has been changed using the server manager administration interface.

RESTRICTIONS

This script must be executed on the machine where the server manager is running.

EXAMPLES

To immediately shut down a server manager named **PoolA** in a server manager configuration named **MyConfig**, enter the following command on a single line:

```
/tc_root/pool_manager/confs/MyConfig_PoolA>mgrstop -Dimmediate
```

rc.tc.mgr_<config>

Shell script to run the server manager for four-tier environments as a Linux daemon. A uniquely-named copy of this script is created in the directory `TC_ROOT/pool_manager/confs/config` by Teamcenter Environment Manager (TEM) for each installed server manager instance. Replace *config* with the server manager name and pool ID. For example, if the server manager name is **MyConfig** and the pool ID is **PoolA**, the script has the following name and location:

`TC_ROOT/pool_manager/confs/MyConfig_PoolA/rc.tc.mgr_MyConfig_PoolA`

To install the daemon, this script must be deployed by the system administrator in the appropriate Linux directory as a postinstallation step. The script is usually launched by the Linux operating system in response to system administration commands.

SYNTAX

`rc.tc.mgr_<config> action`

ARGUMENTS

action

Specifies the daemon control action to be performed:

start

Starts the server manager daemon.

start_msg

Displays a message that the daemon is starting.

stop

Stops the server manager daemon without waiting for servers to become idle.

stop_msg

Displays a message that the daemon is stopping.

status

Displays a message indicating whether the server manager daemon is running.

log

Displays the full output of the server manager.

ENVIRONMENT

None.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

To start a server manager named **PoolA** in a Teamcenter configuration named **MyConfig**, enter the following command on a single line:

```
/tc_root/pool_manager/confs/MyConfig_PoolA>rc.tc.mgr_MyConfig_PoolA start
```

server.mgr_<config>.service

Systemd script and service to run the server manager for four-tier environments as a Linux daemon. The systemd **server.mgr.service** scripts are an alternative to the default `rc.tc.mgr_<config>` script utility.

To install the daemon, this script must be deployed by the system administrator in the appropriate Linux directory as a post-installation step. The script is usually launched by the Linux operating system in response to system administration commands.

During server manager installation on a Linux host, if the **Service / Daemon** startup mode for the server manager is selected, uniquely named versions of the following files are configured and copied to the directory `TC_ROOT/pool_manager/confs/config` for each server manager instance.

File name patterns replace *config* with the server manager name and pool ID. For example, if the server manager name is **MyConfig** and the pool ID is **PoolA**, the script has the following name and location:

`TC_ROOT/pool_manager/confs/MyConfig_PoolA/server.mgr.service_MyConfig_PoolA.service`

server.mgr.service_config.readme

Instructions and scripts for administering the server manager systemd service.

server.mgr_config.service

The server manager systemd service definition.

server.mgr_config.conf

The server manager systemd configuration file. By default, the configuration redirects service log messages to *mgr.output*.

Review the *README* file for operational details on installing the service and using it to administer the server manager daemon.

Subscription Manager

clear_old_newsfeed_messages

Purges messages based on the purge threshold (in number of days) that you set using the **SCM_newsfeed_purge_threshold** preference. Messages that are older than the threshold will be deleted. When the **-process_only_read_messages** option is provided, the utility only considers already read messages for deletion. Unread messages, even those that exceed the threshold, are not deleted.

Note:

This utility can only be run by a Teamcenter administrator.

SYNTAX

```
clear_old_newsfeed_messages -u=user-ID {-p=password | -pf=password-file} -g=group
-process_only_read_messages -h
```

ARGUMENTS

-u

Specifies the Teamcenter administrator's user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the Teamcenter administrator's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-process_only_read_messages

Deletes only read messages.

-h

Displays help for the utility.

purge_invalid_subscriptions

Deletes invalid and expired subscriptions. For security reasons, only a Teamcenter administrator can run this program. Non-administrator users can view the numbers of invalid and expired subscriptions without deleting them.

A subscription references a target object as an external reference. If the target gets deleted, the subscription becomes invalid. The user can interactively delete invalid subscriptions in the rich client interface. Finding a few invalid subscriptions among a large number of subscriptions in the table sometimes is not easy. In addition, subscriptions expire after their expiration dates.

SYNTAX

```
purge_invalid_subscriptions [-u=user-id {-p=password | -pf=password-file} -g=group]
[-report] [-e] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-report

Reports the numbers of invalid and expired subscriptions without deleting them.

-e

Deletes expired subscriptions in addition to deleting invalid subscriptions.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To delete invalid subscriptions, enter the following command on a single line:

```
purge_invalid_subscriptions -u=Tc-admin-user -p=password -g=group
```

- To delete invalid subscriptions and expired subscriptions, enter the following command on a single line:

```
purge_invalid_subscriptions -u=Tc-admin-user -p=password -g=group -e
```

- To report the numbers of invalid and expired subscriptions without deleting them, enter the following command on a single line:

```
purge_invalid_subscriptions -u=Tc-admin-user -p=password -g=group -report
```

System maintenance

clearlocks

Clears dead process locks from the database. Dead process locks typically occur when a Teamcenter session terminates abnormally. Process locks are set on an object when it is being modified or deleted. If a Teamcenter session does not terminate gracefully (by logging out), these locks can remain in place.

Dead process locks (locks held by dead sessions) can cause diverse problems that are often difficult to diagnose, and Teamcenter applications make every effort to eliminate or otherwise avoid them. Nevertheless, there are occasions when such dead process locks must be explicitly removed from the database, and the **clearlocks** utility is used for this purpose.

Note:

To use the **-assert_dead** or **-assert_all_dead** options, you must specify the administrator's user name, password, and group.

The **clearlocks** utility can only obtain general information about the processes in the lock table. Normally, the PID is pulled from the table and a kill is sent to the operating system. If the PID exists, the **Alive** count is incremented. If the PID does not exist, the **Dead** count is incremented. A **Remote PID** count indicates the process was started from a node other than the one that was used to run clearlocks. On some platforms, the kill returns a security violation and no specific information about the PID. If this occurs, the **Other** count is incremented.

SYNTAX

```
clearlocks [-verbose] [-node_names]
[[-assert_dead -u=user-id {-p=password | -pf=password-file} -g=group]
|
[-assert_all_dead -u=user-id {-p=password | -pf=password-file} -g=group]]
[-h]
```

Caution:

The **clearlocks** utility can be run with active Teamcenter sessions, provided that the **-assert_dead** or **-assert_all_dead** arguments are not used. By default, the **clearlocks** utility discriminates between valid and dead process locks; the **-assert_dead** and **-assert_all_dead** arguments defeat this feature.

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument. One of the two mutually exclusive password elements is required.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-verbose

Displays a summary of processes and states (dead, alive, and unknown). Locks associated with dead processes are cleared by the **clearlocks** utility, live processes are not cleared, and the unknown processes are all other processes.

-node_names

Lists nodes upon which the known processes exist.

-assert_dead

Asserts that all processes on a particular node are dead and clears all process locks held by sessions running on that node with the exception of Multi-Site Collaboration transfer locks. If any of those sessions are alive and in use, the locks held by those sessions are compromised.

Note:

To use this argument, you must enter the node name and the administrator's user name, password, and group. To clear Multi-Site Collaboration transfer locks, use the **export_recovery** utility.

-assert_all_dead

Asserts that all processes in the database are dead and clears all process locks with the exception of Multi-Site Collaboration transfer locks. If any of those sessions are alive and in use, the locks held

by those sessions will be compromised. Additionally, this option performs a complete cleanup of the database lock tables, the **POM_TIMESTAMP** table, and reports on the sessions that were asserted to be dead.

Note:

To use this argument, you must enter the administrator's user name, password, and group. To clear Multi-Site Collaboration transfer locks, use the **export_recovery** utility.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- Do not run the **clearlocks** utility with the **-assert_dead** or **-assert_all_dead** arguments if there are any active Teamcenter sessions running. Any locks held by active sessions will be lost and these sessions can then potentially modify data for which they no longer hold modify locks.
- The **-assert_dead** and **-assert_all_dead** arguments are powerful and potentially destructive. Therefore, these arguments should only be used to clear process locks that cannot be cleared otherwise. For this reason, you must enter the administrator's user name, password, and group when using these arguments.
- The **clearlocks** utility cannot clear the **Transfer** lock type, only the **Modify** lock. To clear **Transfer** locks, use the **export_recovery** utility. This behavior is intended to prevent cases where objects that are being transferred are forcibly unlocked and thereby exposing them to the possibility of being modified when their ownership is being transferred.
- When running Clearlocks with the **assert_all_dead** or **assert_dead** option, you may see the message:

```
Notice: There are transfer locks detected indicating active
Multi-Site transfer transactions. All transfers need to complete
before the upgrade can safely continue. Ensure that
ensure_site_consistency is successfully executed for any identified
objects before running Clearlocks.
```

This message also appears when upgrading a database to a new release if there are existing transfer locks in the database.

EXAMPLES

- To clear process locks for dead sessions, enter the following command from that node:

```
$TC_ROOT/bin/clearlocks
```

- To obtain a list of all network nodes which have process locks set on the database, enter the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

- To clear all process locks (active and dead) on a single network node, in this example **ntssun9**, enter the following command:

```
$TC_ROOT/bin/clearlocks -assert_dead -u=Tc-admin-user -p=password -g=group  
ntssun9
```

In this example, *Tc-admin-user* is the administrator's user name and password, and *group* is the administrator's group.

- To clear all process locks (active and dead) on all nodes, enter the following command:

```
$TC_ROOT/bin/clearlocks -assert_all_dead -u=Tc-admin-user -p=password  
-g=group
```

In this example, *Tc-admin-user* is the administrator's user name and password, and *group* is the administrator's group.

- The following is an example of a line message (report) produced by **clearlocks -verbose**:

```
Processes: 7, Alive: 1, Dead: 6, Remote: 0, Other: 0
```

CLEARING PROCESS LOCKS

Perform the following steps to clear dead process locks using the **clearlocks** utility.

1. Ensure that all Teamcenter and Teamcenter Integration for NX users are logged out of the system.

When all users are logged out, all valid process locks are cleared.

2. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

The system displays a report listing network nodes that still have process locks set against the database. Because all users are logged off, these locks are dead and can be cleared.

3. Run the following command:

```
$TC_ROOT/bin/clearlocks
```

4. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

The system displays a report listing network nodes that still have process locks set against the database. Because all users are logged off, these locks are dead and can be cleared.

Any network nodes listed in this second report will require running the **clearlocks** utility with the **-assert_dead** argument to clear the difficult process locks.

5. Run the following command to clear locks held by the session of the specified nodes:

```
$TC_ROOT/bin/clearlocks -assert_dead node-name1 node-name2 node-name3...
```

node-name is a network node listed in the report.

6. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

This report should be clean (empty).

clean_backpointer

Removes **relation_type** object references and **ImanRelation** primary and secondary object references from the backpointer table. As of 2412, these objects are stored in the **ImanRelation** table to improve performance.

Run this utility after upgrading from a version previous to 2412 if your previous deployment stored **relation_type** object references and **ImanRelation** primary and secondary object backpointer references the backpointer table, rather than the **ImanRelation** table.

Before running this utility, ensure that the database is backed up. Running this utility drops any manually created indexes or triggers from the **POM_BACKPOINTER** table. These indexes and triggers must be manually re-created after running the utility.

The utility temporarily creates a copy of the **POM_BACKPOINTER** table while cleaning up the contents of the table. After the copy is created, the original **POM_BACKPOINTER** table is deleted and the copy is renamed to **POM_BACKPOINTER**.

Caution:

Due to renaming of the **POM_BACKPOINTER** table, when running the **clean_backpointer** utility in **CLEAN** mode or **DO_IT** mode, make sure the system administrator has exclusive use of the system.

The default tablespace is used to host this temporary copy of the table if the tablespace argument (**-ts**) is not used. If the **-ts** argument is used, the tablespace specified by the argument is used. Whichever option is used, ensure that the tablespace has enough space to support the size table being created.

Tip:

Run the following to find the size needed for the tablespace:

```
WITH cte1 AS (
  SELECT table_name
    FROM user_tables
   WHERE table_name='POM_BACKPOINTER'
  UNION ALL
  SELECT index_name
    FROM user_indexes
   WHERE table_name='POM_BACKPOINTER'
),
cte2 AS (
  SELECT SUM(blocks)*8/1024 MB
    from dba_extents, cte1
   where owner in (
      SELECT sys_context( 'userenv', 'current_schema' )
        FROM dual
    )
   AND segment_name=cte1.table_name
)
```

```
SELECT 1.25 *SUM(MB)
FROM cte2;
```

If the results of the preceding show that the default tablespace is too small, Siemens Digital Industries Software recommends that you create a new tablespace to hold the modified copy of the **POM_BACKPOINTER** table. Using the size value obtained from the preceding script, create the new tablespace:

```
CREATE TABLESPACE table-space-name ADD datafile 'location\database-file-
name.dbf' size tablespace-sizeG;
```

When you run the utility, specify the new tablespace using the **-ts** argument.

SYNTAX

```
clean_backpointer -u=admin-user-id {-p=password | -pf=password-file} [-g=dba-group]
[-m= INFO | CLEAN | DO_IT]
[-ts=tablespace-name]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument. One of the two mutually exclusive password elements is required.

-g

Specifies the administrative group associated with the user.

If used without a value, the user's default group is assumed.

-m

Specifies whether to provide information about the backpointer table or delete the specified number of objects. Valid values are:

- **INFO**

Provides the **ImanRelation** reference count and a count of **-1** cpids.

- **CLEAN**

Deletes the **ImanRelation** references and fixes **-1** cpids from the backpointer.

- **DO_IT**

Cleans the **POM_BACKPOINTER** table *without* performing the data analysis performed in **CLEAN** mode.

-ts

Specifies the tablespace used to create a modified copy of the **POM_BACKPOINTER** table. This is only for Oracle, and the tablespace must be created beforehand. Ensure that the tablespace is large enough to support the size of the **POM_BACKPOINTER** table. If the utility is run without the **-ts** argument, the default tablespace is used. The new table is moved to the location of the original **POM_BACKPOINTER** table upon successful completion of the utility run after the original **POM_BACKPOINTER** table has been deleted.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To determine how many **ImanRelation** primary and secondary object references exist in your backpointer table, enter the following command on a single line:

```
clean_backpointer -u=Tc-admin-user -p=password -g=group -m=INFO
```

The utility returns the total number of backpointer entries, as well as the number of **ImanRelation** objects identified for deletion.

- To delete all **ImanRelation** primary and secondary object references from your backpointer table, enter the following command on a single line:

```
clean_backpointer -u=Tc-admin-user -p=password -g=group m=CLEAN
```

- To rebuild the **POM_BACKPOINTER** table, enter the following command on a single line:

```
clean_backpointer -u=Tc-admin-user -p=password -g=group m=DO_IT
```

cleanup_recovery_table

Locates and removes all recipe objects resulting from time-out sessions and client crashes. (Teamcenter recipe objects are used to facilitate transparent recovery when a **TcServer** instance crashes.) Typically, recipe objects are deleted when users log off. But during client crashes or session time-outs, recipe objects may remain in the database if the session or objects are not recovered. Run this utility periodically to clear the database of recipe objects.

SYNTAX

```
cleanup_recovery_table -u=user-id {-p=password | -pf=password-file} [-g=group]
[-age=number-of-days] [-report=output-file] [-verbose] [-dryrun] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-age

Specifies the age (in days) recipe objects must reach to be eligible for deletion. Recipe objects older than the specified age are deleted when the utility is run.

By default, the age is processed as **30** if you do not specify a value for this argument.

-report

Specifies the name of the output report file to which a summary of deleted recipe objects are logged. The summary states how many recipe objects were deleted, from how many dead sessions.

Accepts the relative or full path and file name.

-verbose

Logs a complete list of all deleted recipe objects (as opposed to a summary) to the file specified by the **-report** argument.

If the **-report** argument is not specified, the report is output to the console.

-dryrun

Generates a report stating how many recipe objects exist in the database for how many dead sessions. No recipe objects are deleted.

Siemens Digital Industries Software recommends running this argument first, before actually deleting recipe objects from the database.

If this argument is run with the **-age** argument, the report states how many recipe objects of the specified age exist in the database, relative to the total number of recipe objects.

If the **-report** argument is not specified, the report is output to the console.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To delete all recipe objects older than 30 days and write a summarized report of the deleted objects to the **deletedrecipes.txt** file, enter the following command on a single line:
`cleanup_recovery_table -age=30 -report=deletedrecipes.txt`
- To delete all recipe objects older than 10 days and write a full report of all deleted objects to the **deletedrecipes.txt** file, enter the following command on a single line:
`cleanup_recovery_table -age=10 -report=deletedrecipes.txt -verbose`
- To test the removal of all recipe objects older than 30 days and write a summarized report of the deleted objects to the **deletedrecipes.txt** file, enter the following command on a single line:
`cleanup_recovery_table -age=30 -report=deletedrecipes.txt -dryrun`

generate_client_meta_cache

Note:

The generate_client_meta_cache utility is deprecated in Teamcenter version 13.2. It is superseded by **bmid_generate_client_cache**.

Generates the shared client metadata cache. This a group of datasets in the Teamcenter database that are organized within the **ClientCache** folder under Teamcenter administrator's home folder. These datasets can contain any types of files, for example: **zip** files, **png** files, and **xml** files.

This utility is run during the installation and upgrade, and can be run manually as desired.

Following are some of the situations when you should run this utility:

- Generate the full client cache (**all** argument) if the client cache folder has been deleted.
- Generate the style sheet cache (**stylesheets** argument) after a style sheet is created or edited. This keeps the style sheet feature in the client cache folder up-to-date.

For more information, see *Rich Client Customization*.

- Generate the text server cache (**textservers** argument) after text server data has been updated.

For more information about working with text server files, see *Teamcenter Localization*.

Metadata is cached on the client computer, eliminating server calls for the cached metadata. When the rich client connects to the server, it requests the client metadata cache files from the File Management System (FMS), which automatically downloads, refreshes, and maintains local copies as needed.

The following are some features of the client metadata caching functionality:

- Caching refresh notification

Notifies you when the cache is refreshed. When you log on to the rich client, a check is made for new metadata on the server. If new metadata is found, the cache on the rich client is refreshed, and the following message is displayed:

Synchronizing the Rich Client install files with the Teamcenter Server.

- The **ClientCache** Teamcenter folder

Stores cached metadata. If you are logged on to the rich client as the Teamcenter administrator who generated the cache, you see the folder under the **Home** location. There are folders which organize datasets into categories. These datasets contain the individual files.

Warning:

If the administrator moves, renames, or deletes this folder, all clients fall back to a nonclient-cached mode. This folder name, and all those under it must be unique within Teamcenter.

- **Generate Client Cache** check box

Runs the **generate_client_meta_cache** utility. The check box appears in Teamcenter Environment Manager (TEM) and the Business Modeler IDE.

For more information, see *Teamcenter Environment Manager — Usage and BMIDE for Data Model Design*.

- **TC_SKIP_CLIENT_CACHE**

Causes the rich client to run in legacy mode and avoid downloading or using a local client meta cache.

For more information, see the *Teamcenter Environment Variables*.

SYNTAX

```
generate_client_meta_cache [-u=Tc-admin-user {-p=password | -pf=password-file} -g=group]
command [generate | update | delete | report | tickets | context]
cache [all | types | lovs | stylesheets | textservers]
-log=log-file-path
-delta=differences-file-path
l1=language
l2=language
-err=error-file-name
-t=timings
-remote
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against

the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

command

Specifies the command to run. (You must run the command as a user with Teamcenter administration privileges.)

generate

Generates the client cache.

update

Updates the client cache as needed. This is the default setting.

delete

Deletes the client cache.

report

Reports on the state of the client cache.

tickets

Generates tickets for the feature.

context

Checks context for LOVs.

cache

Specifies the client cache to work on.

all

Generates all the available client cache files. This is the default setting.

types

Generates the types cache. This includes the metadata of business objects, property descriptions, and lists of values.

lovs

Generates the list of values (LOVs) cache.

icons

Generates the icons cache.

stylesheets

Generates the stylesheets cache.

textservers

Generates the text server cache.

tools

Generates the tools cache.

contextmenusuppressionrules

Generates the context menu suppression rules cache.

-log

Specifies file path and name of the log file that contains the results of this execution. The default log file is written to the **TEMP** directory.

This argument is optional.

-delta

Specifies the file path and name of the file into which data model differences are written.

-l1

Specifies the first language for which to generate the client cache, for example, **en_US**, **de_DE**, **es_ES**, **fr_FR**, **pt_BR**.

-l2

Specifies the second language for which to generate the client cache.

-err

Specifies the error file name. The default error file is written to the **TEMP** directory.

-t

Specifies time function calls.

-remote

Specifies remote start. This generates log files.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To generate or regenerate all the client meta cache, enter the following command on a single line.

```
generate_client_meta_cache -u=Tc-admin-user -p=password -g=group generate
all
```

This command keeps the previous version of cache data in the server.

- To generate style sheet client meta cache, enter the following command on a single line.

```
generate_client_meta_cache -u=Tc-admin-user -p=password -g=group generate
stylesheets
```

- To generate text server files client meta cache, enter the following command on a single line.

```
generate_client_meta_cache -u=Tc-admin-user -p=password -g=group generate  
textservers
```

bmide_generate_client_cache

Generates the shared client metadata cache. This is a group of datasets in the Teamcenter database that are organized within the **ClientCache** folder under the administrator's home folder. These datasets can contain any types of files, for example: **zip** files, **png** files, and **xml** files.

This utility is run during the installation and upgrade, and can be run manually as desired. Following are some of the situations when you should run this utility:

- Generate the full client cache (**all** argument) if the client cache folder has been deleted.
- Generate the text server cache (**textservers** argument) after text server data has been updated.

For more information about working with text server files, see *Teamcenter Localization*.

Metadata is cached on the client computer, eliminating server calls for the cached metadata. When the rich client connects to the server, it requests the client metadata cache files from the File Management System (FMS), which automatically downloads, refreshes, and maintains local copies as needed.

The following are some features of the client metadata caching functionality:

Caching refresh notification	Notifies you when the cache is refreshed. When you log on to the rich client, a check is made for new metadata on the server. If new metadata is found, the cache on the rich client is refreshed, and the following message is displayed: Synchronizing the Rich Client install files with the Teamcenter Server.
The ClientCache Teamcenter folder	Stores cached metadata. If you are logged on to the rich client as the Teamcenter administrator who generated the cache, you see the folder under the Home location. There are folders which organize datasets into categories. These datasets contain the individual files. Warning: If the administrator moves, renames, or deletes this folder, all clients fall back to a nonclient-cached mode. This folder name, and all those under it must be unique within Teamcenter.
Generate Client Cache check box	Runs the bmide_generate_client_meta_cache utility. The check box appears in Teamcenter Environment Manager (TEM) and the Business Modeler IDE. For more information, see <i>Teamcenter Environment Manager — Usage</i> and <i>BMIDE for Data Model Design</i>.
TC_SKIP_CLIENT_CACHE	Causes the rich client to run in legacy mode and avoid downloading or using a local client meta cache.

For more information, see the *Teamcenter Environment Variables*.

SYNTAX

```
bmide_generate_client_cache [-u=Tc-admin-user {-p=password | -pf=password-file} -g=group]
-mode=[generate | update | delete]
-model_file=model-file-path
-target_dir=target-directory-path
-upload=[true | false]
-cache=[all | types | lovs | types_loc | lovs_loc | icons | stylesheets | tools | textservers |
contextmenusuppressionrules]
-locales=list-of-locales
-delta=delta-file-path
-log=log-file-path
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies the operation to perform on the client cache.

generate Generates the client cache.

update Updates the client cache as needed.

delete Deletes the client cache.

-model_file

Specifies the model file from which to generate the client cache.

-target_dir

Specifies the output path of generated client cache files.

-upload

Specifies if the client cache to be uploaded. This is an optional argument

-cache

Specifies the client cache to work on. This is an optional argument.

all Generates all the available client cache files. This is the default setting.

types Generates the types cache. This includes the metadata of business objects, property descriptions, and lists of values.

lovs Generates the list of values (LOVs) cache.

types_loc Generates the localization for types. By default generates for all locales.

lovs_loc Generates the localization for LOVs. By default generates for all locales.

icons Generates the icons cache.

stylesheets Generates the stylesheets cache.

tools Generates the tools cache.

textservers Generates the textservers cache.

contextmenusuppressionrules Generates the context menu suppression rules cache.

-locales

Specifies a list of comma separated locales for which to generate the client cache. For example, **"en_US,de_DE"**. This argument is optional. If the locales argument is not specified or no value is specified, then localizations are generated for all locales.

Example languages:

en_US English (United States)

de_DE German

es_ES Spanish

fr_FR French

it_IT Italian

pt_BR Portugese (Brazil)

cs_CZ Czech

pl_PL Polish

ru_RU Russian

ko_KR Korean

ja_JP Japanese

zh_TW Chinese (Taiwan)

zh_CN Chinese (PRC)

-delta

Specifies the file path and name of the file into which data model differences are written. The delta argument is mandatory if **-mode=update**.

-log

Specifies file path and name of the log file that contains the results of this execution. The default log file is written to the **TEMP** directory.

This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To generate or regenerate all the client meta cache, enter the following command on a single line.

```
bmid_generate_client_cache -u=Tc-admin-user -p=password -g=group  
-mode=generate -cache=all -model_file=%TC_DATA%/model/model.xml  
-target_dir=%TC_DATA%/clientcache
```

This command keeps the previous version of cache data in the server.

- To update the client meta cache, enter the following command on a single line.

```
bmid_generate_client_cache -u=Tc-admin-user -p=password -g=group  
-mode=generate -cache=all -model_file=%TC_DATA%/model/model.xml  
-target_dir=%TC_DATA%/clientcache -delta=%TC_DATA%/model/delta.xml
```

generate_metadata_cache

Generates the metadata shared server cache dataset. The cache is generated only if constant, type, or property metadata is updated since the last update. The **-force** argument forces regeneration of the cache.

If you receive the following message, an administrator must run this utility on the server:

The schema file is out of date. Please regenerate.

This indicates that when a template containing schema changes was installed to the server, the **Generate Server Cache** check box was not selected in Teamcenter Environment Manager (TEM) or the Business Modeler IDE. (It is also possible that the error message appears because the server instance the user is attempting to connect to does not have an up-to-date server cache. The user should log off and log on several times to connect to a fresh server instance. If the error message persists, the user should contact the server administrator to regenerate the server cache.)

For more information about managing shared server cache, see *Teamcenter Administration*.

ARGUMENTS

-u

Specifies the user ID.

Specifies the user ID of the user with administrative privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-force

Forces creation of the dataset even if a correct version already exists.

-delete

Deletes the dataset.

-log

Specifies file path and name of the log file that contains the results of this execution.

This argument is optional.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

install_event_types

Note:

This utility and the **define_auditdefs** utility are deprecated and will be removed in a future version.

Beginning in Teamcenter 10, the formerly available **create** parameter used with the **-f** argument is obsolete, and therefore you can no longer create event types using this utility. Use the Business Modeler IDE to create custom event types.

For more information, see *BMIDE for Data Model Design*.

Defines which event and object types can be subscribed to and/or audited. This utility also adds event types to an object type.

SYNTAX

```
install_event_types [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=function {install | add | remove | modify | listValidEvents
| listEventtypes | text-file-name} [-overwrite]
[-eventtype=eventTypeId] [-imantype=imanTypeName]
[-imanclass=imanClassName] [-remove] [-audit] [-noaudit]
[-subscribe] [-nosubscribe] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with **DBA** privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the mode in which the utility executes. The mode must be one of the following:

- **install**

Installs standard event types and event type mappings.

- **add**

Adds the event type to the specified imantype and imaclass.

- **remove**

Removes event types or event type mappings.

- **modify**

Modifies event type mapping.

- **listValidEvents**

Lists valid event types for the specified imaclass and imantype.

- **listEventtypes**

Lists all the event types in the database.

- *text-file-name*

Specifies the file name used to create event type mappings, modify event type mappings, or delete event types and event type mappings.

-overwrite

Overwrites existing definitions, if any, during installation. Valid only with the **-f=install** and **-f=input-file** arguments.

-eventtype

Specifies the event type ID when creating or deleting event types or event type mappings.

-imantype

Specifies the **imantype** name when creating or deleting event types or event type mappings or when listing valid event types for that particular **imantype**.

-imaclass

Specifies the **imaclass** name when creating event type mappings, deleting event type mappings, or listing valid event types for that particular **imaclass**.

-remove

Use with the **-f=file-name** option to perform the remove operation (remove event types and event type mapping) on the file data.

-audit

Specifies that the event type can be audited. Valid only with the **-f=modify** argument.

-noaudit

Specifies that the event type cannot be audited. Valid only with the **-f=modify** argument.

-subscribe

Specifies that the event type can be subscribed to. Valid only with the **-f=modify** argument.

-nosubscribe

Specifies that the event type cannot be subscribed to. Valid only with the **-f=modify** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

When deleting event types and event type mappings using a file, all event type mappings for the event type must be deleted prior to deleting event type itself. For more information, see the example for using the **-f=file-name -remove** option, below.

EXAMPLES

Note:

Adding a custom event type requires implementing a **CUSTOM_event_handler**.

- To install the default event types and event type mapping definitions, enter the following command on a single line:

```
install_event_types -f=install -overwrite
```

The **-override** switch causes the install to overwrite any existing definitions.

- To add the **MyEventType** event type as a valid event type for the Teamcenter type **Item**, enter the following command on a single line:

```
install_event_types -f=add -imantype=Item -imaclass=Item  
-eventtype=MyEventType
```

- To remove the **MyEventType** event type from the Teamcenter type **Item**, enter the following command on a single line:

```
install_event_types -f=remove -imantype=Item -imaclass=Item  
-eventtype=MyEventType
```

Note:

Users cannot remove Teamcenter internal event types.

- To set the **MyEventType** event type as subscribable but not auditable for the Teamcenter type **Item**, enter the following command on a single line:

```
install_event_types -f=modify -imantype=Item  
-imaclass=Item -eventtype=MyEventType -audit -nosubscribe
```

- To list all valid event types for the Teamcenter type **Item**, enter the following command on a single line:

```
install_event_types -f=listValidEvents -imantype=Item -imaclass=Item
```

- To list all event types defined in the database, enter the following command on a single line:

```
install_event_types -f=listEventtypes
```

- To read the specified file, define new event types to install, (each line with an event type name) and defines event type mappings (each line with an event type mapping) in the following format:

```
iman_type_name, iman_class_name, event_type_name, subscribable_flag,  
auditable_flag  
install_event_types -f=C:\temp\my_event_types.txt -overwrite
```

In the **C:\temp\my_event_types.txt**:


```

my_event_type_1
my_event_type_2
EngChange,Item,my_event_type_1,true,true
EngChange,Item,my_event_type_2,true,false
EngChange Revision,ItemRevision,my_event_type_1,true,true
EngChange Revision,ItemRevision,my_event_type_2,true,false

```

- To read the specified file, delete event type mappings, (each line with an event type mapping) and delete event types (each line with an event type name) in the formats shown, enter the following command on a single line:

```
install_event_types -f=C:\temp\my_event_types.txt -remove
```

Format to delete event type mapping

```
iman_type_name, iman_class_name, event_type_name
```

Format to delete event type

```
event_type_name
```

The **my_event_types.txt** file contains the following:

```

EngChange,Item,my_event_type_1
EngChange,Item,my_event_type_2
EngChange Revision,ItemRevision,my_event_type_1
EngChange Revision,ItemRevision,my_event_type_2
my_event_type_1
my_event_type_2

```

- To delete the **MyEventType** event type, enter the following command on a single line:

```
install_event_types -f=remove -eventtype=MyEventType
```

To set display names for event types:

- Open the **user_property_names.xml** file. This file is located in the *TC_ROOT\lang\textserver\en_US* directory.
- Add the keys for the event names as follows:

```

<key id="k_et_event1">Event Name 1</key>
<key id="k_et_event2">Event Name 2</key>

```

- Stop the pool manager.
- Delete the Teamcenter shared memory files.

The shared memory files are typically located in the temporary files location. For example, in Windows, the shared files are located in the **C:\Temp***P_folder number* directory.

- Restart the pool manager.

list_types

Lists all types in the Teamcenter database. Use the output of this utility as input to the [database_verify](#) utility for the offline case.

SYNTAX

```
list_types [-u=user-id {-p=password | -pf=password-file} -g=group]
[-outfile=output-file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID of the user with administrative privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-outfile

Specifies the name of the file to contain the output. If this argument is not specified, all types information is displayed on the console.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

list_users

Creates a list of users currently logged on to Teamcenter and the node they are using. (The list of users includes the user running the utility.) This information is useful if database maintenance is necessary and all users currently logged on must be notified.

SYNTAX

list_users [-u=user-id {-p=password | -pf=password-file} -g=group] -notkilldeadprocesses [-h]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-notkilldeadprocesses

Disables clearing of dead process locks.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

migrate_home_folder

Finds the instances of the home folder for all users in the Teamcenter database and modifies the **object_type** property value to **Fnd0HomeFolder**. The **Fnd0HomeFolder** business object represents the home folder and is a child of the **Folder** business object.

This utility is run automatically during upgrade. You only need to run it manually if the utility did not complete during upgrade.

By default, this utility finds and process only the unmigrated instances. You can force it to process the objects that are already migrated by specifying the **-force_rerun** argument. You can rerun the utility any number of times. You can process all the home folder instances in bulk using the **-objects** argument.

SYNTAX

```
migrate_home_folder -u=user-ID {-p=password | -pf=password-file} [-g=group]
[-objects=number-of-objects-per-iteration]
[-force-rerun]
[-log=log-file-path]
[-v]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have administrative privileges.

-p

Specifies the user's password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-objects

Specifies the number objects to be processed per iteration. The default value is 1000 objects. The input value must be a positive integer. If the number of objects specified exceeds the maximum objects in the database, this utility processes all instances in the first iteration itself.

-force_rerun

Forces the utility to query and migrate the instances that are already migrated along with unmigrated instances.

-log

Specifies the path for the log file. The log file contains the information about the number objects successfully migrated, the number of objects not migrated, the time taken to complete the operation, and so on. If this argument is not specified, the log file is generated under the directory defined by the **TC_TMP_DIR** environment variable.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

Apart from the default log files specified in *Log files produced by Teamcenter*, this utility generates an additional log file with the details of the number of instances migrated, the number of instances that failed to migrate, and the time taken to complete the operation. The additional log file has this format:

migrate_home_folderYYYY_MM_DD_HH_MM_SS.log

You can find it in the directory specified by the **TC_TMP_DIR** environment variable.

RESTRICTIONS

You must be a Teamcenter administrator to execute this utility.

EXAMPLES

- To migrate all home folder instances in the database with up to 5000 objects per iteration, enter the following command.

```
migrate_home_folder -u=Tc-admin-user -p=password -g=group -objects=5000
```

- To migrate all home folder instances in the database with up to 10000 objects per iteration and to force the utility to rerun the migration on already migrated instances along with unmigrated instances:

```
migrate_home_folder -u=Tc-admin-user -p=password -g=group -objects=5000 -force_rerun
```


prune_named_references

Finds, removes, and deletes named references from associated datasets. Associated datasets containing no named references after the prune operation are deleted.

The named references can be obtained from saved query results.

A provided saved query, **ImanFileNamedRefsQ**, uses the following search criteria:

- The **Original File Name** attribute is the **original_file_name** of the **ImanFile** file.
- The **Last Modified Date** attribute is the **last_modified_date** of the **ImanFile**.
- The **Name** value is the type name of the associated dataset.
- The **Type** value is the type name of the associated business object related to the dataset with a **Specification** relationship.

You can use **prune_named_references** to perform the following tasks:

- Query the named references through an existing saved query.

The query input contains the name of the existing saved query, the name-value map of the entry named, and the value of the saved query. The query output contains the candidates of the named reference to be pruned in an XML file with the **UID**, **original_file_name**, **last_modified_date**, size of the file stored in volume, and the class name of the named references.

- Prune the named references.

The input candidates to be pruned are stored in an XML file that has the same format as the query results. You can edit the query results file in a Microsoft Excel file and use that as the input to the prune operation.

- Query and prune.

The utility queries the candidates using the query information and then prunes all the candidates from the query results.

Syntax

```
prune_named_references -u=user-id {-p=password | -pf=password-file} [-g=group]
[-encrypt={true | false}]
[-mode={prune | query | all}]
[-class=class_name ]
[-savedQueryInfo=savedQueryInfoFile -queryOut=queryResultsFile] | [-
```

pruneNamedRefs=pruneInputFile]
[-h]

Arguments

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-encrypt

If set to **true**, specifies password encryption.

-mode

query	Query the named references with the specified savedQueryInfo file and write the results to the queryOut file.
prune	Prune the named references in file specified by the pruneNamedRefs file.
all	Query the named references with savedQueryInfo file and prune all the named references obtained from query results

-class

Specifies the class name of the candidates for pruning; the default class name is **ImanFile**.

-savedQueryInfo

Specifies the required XML file **query** and **all** modes; contains the saved query name and the name value map. An example input file follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- existing saved query name -->
<savedQuery name="ImanFileNamedRefsQ">
  <!-- attribute original_file_name of the ImanFile -->
  <entry name="Original File Name" value="*.op2"/>
  <!-- attribute late_mod_date of the ImanFile [optional] -->
  <entry name="Last Modified Date" value="04-Sep-2018 16:38"/>
  <!-- owning Dataset Type name of the ImanFile -->
  <entry name="Name" value="CAESolution"/>
  <!-- owning Item Revision Type name of the Dataset -->
  <entry name="Type" value="CAEAnalysisRevision"/>
</savedQuery>
```

-queryOut

Specifies the XML file to contain the named references from the query results.

-pruneNamedRefs

Specifies the XML file to contain the named references to be pruned.

-h

Displays help for this utility.

Environment

As specified in *Setting up a Teamcenter command-line environment*.

Files

As specified in *Log files produced by Teamcenter*.

Restrictions

You must be a Teamcenter administrator to execute this utility.

Examples

- Query candidates:

```
prune_named_references -u=admin_id -p=admin_password
-pf -g=admin -mode=query -savedQueryInfo=d:\myWork\queryInfo.xml
-queryOut=d:\myWork\queryOut.xml
```

- Run the prune operation:

```
prune_named_references -u=admin_id -p=admin_password  
-pf -g=admin -mode=prune - pruneNamedRefs =d:\myWork\queryOut.xml
```

- Query for candidates and prune the candidates:

```
prune_named_references -u=admin_id -p=admin_password  
-pf -g=admin -mode=all - savedQueryInfo=d:\myWork\queryInfo.xml
```

reset_user_home_folder

Repairs corruption that may occur when deleting a user from the database by redirecting the user's home folder, mailbox folder, and **Newstuff** folder to the home folder of the administrator running this utility.

If a home folder owned by the administrator is found, the utility points the deleted user's home folder back the administrator's home folder. If the administrator's home folder is not found, the utility creates a new home folder and redirects the deleted user's home folder to it.

SYNTAX

```
%TC_BIN%\reset_user_home_folder [-u=user-id {-p=password | -pf=password-file}
-g=group] -id=user-id-whose-home-folder-is-to-be-reset [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-id

Specifies the user ID of the user who owns the home folder to be reset.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

site_util

Performs site-related maintenance, such as creating and deleting remote sites, setting ownership, and changing the ID of remote sites.

SYNTAX

```
site_util [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=function [-site_name=site-name] [-site_id=site-id]
[-license_server=license-server] [-node_name=node-name] [-soa_url=soa-url]
[-gms_url=gms-url] [-geography=geography_code]
[-ods=y | n] [-hub=y | n] [-http=y | n] [-tcxml=y | n] [-offline=y | n]
[-replicaDel=y | n] [-unmanaged=y | n] [-archive=y | n] [-display_only] [-h]
```

ARGUMENTS

-u

Specifies the user ID. This must be a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

create

Defines a remote site in the local database.

set_id

Changes the current ID of a remote site. This function corrects errors made while defining sites and must be used with extreme caution.

Caution:

Modifying a site ID can result in serious data sharing problems. This function is not allowed to be run on the local site, because it will corrupt the database.

The **-site_name** argument is required when using the **set_id** function.

modify

Changes attributes of a site other than the site ID and may be used for local or remote sites. The **-site_id** argument is required to identify the site being modified. Only the given attributes are modified.

delete

Deletes a remote site from the local database. Only those sites not referenced by any object in the local database can be deleted. This function cannot be used for the local site. The **-site_id** argument is required when using the **delete** function.

list

Lists all defined sites in the database and their attributes. No attribute switch is required.

fix_site_ownership

Sets the ownership of all defined sites in the database as being owned by the local site. Use this when you encounter an error stating that you cannot export a **POM_imc** object because it is owned by another site. This can be run while users are logged on to the database. No attribute switch is required.

-site_name

Specifies the name of the site when creating or modifying sites.

-site_id

Specifies the ID of the site to which the specified function applies.

-license_server

Specifies a default license server for the site. If, when creating a user, if you do not assign the user to a specific license server, the default license server is used.

-node_name

Specifies the node name when creating a new site. This argument can also be used when modifying site properties. If the value of the **-http** argument is set to **y**, the value of **-site_name** can be a URL of an SOA.

-soa_url

Specifies the service oriented architecture (SOA) URL, for example, **http://Tc_host:7001/tc**.

-gms_url

Specifies the URL for Global Services.

-geography

Specifies the geography code, a two-character code from ISO 3166.

-ods

Specifies whether the site being created or modified is an Object Directory Services (ODS) site. For more information about Object Directory Services, see *Multi-Site Collaboration*.

-hub

Specifies whether the site being created or modified is a hub. For more information about hub configurations, see *Multi-Site Collaboration*.

-http

Specifies whether to use the HTTP protocol or remote procedure call (RPC) protocol.

-tcxml

Specifies the utility uses TC XML payload instead of an object manager.

-offline

Specifies that the site has no network connection to the local site. This argument is intended to avoid sending unnecessary messages to a site when a replica is deleted.

-replicaDel

Switch that indicates objects replicated to a given site can be deleted as long as the object is not replicated to other sites that do not have this property.

-unmanaged

Switch that indicates an unmanaged site.

-archive

Specifies whether the site being created or modified is a Multi-Site Collaboration archive site. Only a single site in a federation of connected sites can be specified as an archive site.

-display_only

Determines whether it is necessary to fix site ownership. This argument must be used in conjunction with the **fix_site_ownership** function. A returned count greater than zero indicates that site ownership must be fixed by running the utility using the **fix_site_ownership** function.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To define a new remote site in the local database, enter the following command on a single line:

```
site_util -f=create -site_id=123456789 -site_name=Site1
-node_name=node1 -ods=y
```

- To change the node name of a site, enter the following command on a single line:

```
site_util -f=modify -site_id=123456789 -node_name=node2
```

- To indicate that a site is a multisite hub, enter the following command on a single line:

```
site_util -f=modify -site_id=123456789 -hub=y
```

A returned count greater than zero indicates that site ownership must be fixed by running the utility using the **fix_site_ownership** function.

- To determine if it is necessary to run the **fix_site_ownership** function, enter the following command on a single line:

```
site_util -f=fix_site_ownership -display_only
```

- To define a GMS site, enter the following command on a single line:

```
site_util -f=create -http=y -tcxml=y
-node_name=http://url_of_the_gs_instance
```

runBatch

Script that allows a site to execute a group of translations together. The translations are contained in either the default **testxmlTranslationPerf.xml** file or any other XML file and the path of the XML file must be passed as argument.

SYNTAX

```
-u=user-id {-p=password | -pf=password-file} [-g=group]
TranslationManagementRootDir/AdminClient/bin/runBatch
[-help] input-batch-xml-file-path
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

input-batch-xml-file-path

Input batch XML file path. This argument is optional. If the path is not specified, the system uses the default path as `..\testxml\TranslationPerf.xml`.

-help

Displays help for this utility.

ENVIRONMENT

None.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

- The following is an example of a translation XML file:

```
<?xml version="1.0"?>

<!--
This software and related documentation are proprietary to UGS Corp.
COPYRIGHT 2007 UGS CORP.  ALL RIGHTS RESERVED
-->

<TranslationTasks>
  <!-- RootDir is the common directory for all the Input files in this
xml file.>
  <  This can be absolute path or relative path from AdminClient/bin
Directory.>
  <RootDir value="../data"/>
  <!-- To add more tasks to the batch, copy and insert more
TranslationTask
elements >
  <TranslationTask Submits="1" Provider="UGS" Service="tozipfile"
context="Translation">
    <Priority value="2"/>
    <!-- For time based tasks uncomment Time tag.
Time format is "MM/dd/yyyy HH:mm" -->
    <!--Time value="01/25/2007 17:33"/-->
    <Options>
      <Option key="Test_Key" value="Test_Value"/>
    </Options>
    <!-- If RootDir is specified input file path is
```

```

value of "RootDir + Input". >
  <    If RootDir is not specified input file path is
value of "Input" and    >
  <    has to be absolute path.                                >
  <    This applies to the both Input and Dependant element attribute.--
>
  <Input value="ssw_idi0001.idi"/>
  <!-- Dependant values take * wildcard -->
  <!-- Dependant value="*.dep"/ -->
  </TranslationTask>

</TranslationTasks>

```

- The following are examples of command line entries:

```

runbatch test test
runbatch test test ..\testxml\TranslationPerf.xml

```

tc_mail_smtp

Sends SMTP (Simple Mail Transfer Protocol) e-mail. Use this platform-independent utility when sending e-mail on both Linux and Windows platforms.

E-mail server settings are specified using Teamcenter Environment Manager during Teamcenter installation as described in *Advanced Foundation settings* in the *Teamcenter Environment Manager Help*.

Caution:

The only authentication for this tool is performed by the operating system. Therefore, you must take precautions to ensure that the utility is not misused. Use this utility at your own risk.

SYNTAX

```
tc_mail_smtp
{-to=address | -to_list_file=email-list-file-name}
[-cc=address]
[-bcc=address]
[-subject=subject-line]
[-server=mail-server-name [:port]]
[-port=port-num]
[-body=file-name[-body=alternate-file-name]]
[-attachments=file-name]
[-user=sender's-name]
[-name=display-name]
[-charset=character-set]
[-validation=validation-mode]
[-testmode=file-name]
[-priority={1 | 2 | 3}]
[-auth={true | false}]
[-authId=authenticating-ID]
[-pf=qualified-file-name]
[-conSecurity={None | SSL/TLS | STARTTLS}]
[-sslTlsProtocol={TLSv1 | TLSv1_1 | TLSv1_2 | TLSv2 | TLSv23 | TLSv3}]
[-h]
```

ARGUMENTS

-to

Specifies the e-mail address of the recipient. If there are multiple recipients, use a **-to** argument for each recipient.

Either this argument or the **-to_list_file** argument is required.

-to_list_file

Specifies the full path and file name of the file containing the list of e-mail addresses. Each line of the file must contain a full e-mail address.

Either this argument or the **-to** argument is required.

-cc

Specifies the e-mail address of the carbon-copied recipient. If there are multiple recipients, use a **-cc** argument for each recipient.

-bcc

Specifies the e-mail address of the blind carbon-copied recipient. If there are multiple recipients, use a **-bcc** argument for each recipient.

-subject

Specifies the subject line of the e-mail. Accepts a string up to 100 characters. Enclose the string text in double quotes.

-server

Specifies the name of the mail server. Accepts a string up to 100 characters. Optionally, you can define the mail server port within this argument. For example:

```
-server=myserver:1234
```

Alternatively, you can define the mail server port using the **-port** argument. If you do not use either method to specify the mail server port, the SMTP mail port (**25**) of the local machine is used.

-port

Specifies the mail server port. If not defined, the SMTP mail port (**25**) of the local machine is used.

-body

Specifies the full path (limited to 200 characters) to the file containing the text of the e-mail body. Limited to 200 characters. You can also specify an alternate file name, in a different format. Files can be either text files or HTML files. (HTML files extensions can be only **.htm** or **.HTM**.) For example:

```
-body=C:\mail\body\filename1.txt
-body=C:\mail\body\filename2.htm
```

You can use this argument to specify a total of one text file, or one HTML file, or one text file and one HTML file. You cannot specify, two or more text files, or two or more HTML files.

If you specify one text file and one HTML file, the e-mail always contains the text message first, and then the HTML message, regardless of the order in which you sent the argument values.

If you specify multiple values for the same body type, the last value entered in the command line for this argument is read by the system. For example, if you specify:

```
-body=C:\mail\body\filename1.txt
-body=C:\mail\body\filename2.txt
-body=C:\mail\body\filename3.txt
```

Only the contents of **filename3.txt** are added to the body of the message.

-attachments

Specifies the full path and file name of one or more attachment lists and the format of each attachment (**B**=binary text, **T**=text). The full path and file name cannot exceed 200 characters.

For example, you can specify the full path of the location of the **bin_file** file and indicate the file is a binary file:

```
C:\mail\attachments\attachment1\bin_file=B
```

To include graphics within the HTML body of an e-mail, specify the image source within the HTML (**img src=cid:myimage.jpg**), and then specify the same image source within the attachment file. For example:

```
C:\mail\attachments\attachment1\myimage.jpg=B
```

To include several graphic attachments with the same name, differentiate the files with IDs. In the HTML body of an e-mail, specify the various image sources within the HTML (**img src=cid:fred_image.jpg**) and (**img src=cid:bob_image.jpg**). Then specify the corresponding image sources within the attachment file. For example:

```
C:\mail\attachments\attachment1\myimage.jpg=B,html,id=fred_image.jpg
C:\mail\attachments\attachment2\myimage.jpg=B,html,id=bob_image.jpg
```

-user

Specifies the name of the user from whom the e-mail is sent. If not defined, the login name of the local machine is used.

-name

Specifies the name as presented to the recipient.

-charset

Specifies the character set of the message body. Possible values are defined by the Internet Assigned Numbers Authority (IANA).

-validation

Determines the behavior of e-mail delivery if an invalid e-mail address is entered.

0	E-mail delivery continues, even if incorrect e-mail addresses are encountered. No error is returned. This is the default setting.

- 1 E-mail delivery continues, even if incorrect e-mail addresses are encountered. An error is returned.
- 2 E-mail delivery is sent first to recipients on the **To** list, then the **CC** list, and then the **BCC** list. Delivery stops at the first invalid address. An error is returned.

-testmode

Allows you to print a test of the e-mail output. Specify either **stderr**, **stdout**, or the full path and file name to the **stderr** file.

-priority

Specifies the priority of the e-mail.

- 1 The e-mail is sent with high priority and high importance.
- 2 The e-mail is sent with normal priority.
- 3 The e-mail is sent with low priority and low importance.

-auth

Specifies whether authentication is used when Teamcenter connected to a mail server.

- false** (The default value.) E-mail sent from Teamcenter is not authenticated.
- true** E-mail sent from Teamcenter is authenticated using the values specified with **-authId** and **-pf**.

-authId

Specifies the e-mail ID or the user ID to use for e-mail authentication. Chose an e-mail ID (such as **teamcenter@yourdomain.com**) or user ID (such as **tcserver01**) based on your e-mail server requirements.

-authId is used when **-auth** is set to **true**.

pf

Specifies the full path and file name of the encrypted password to use for e-mail authentication. For example,

```
"C:\Teamcenter\TcSecurity\email_password"
```

-pf is used when **-auth** is set to **true**.

The encrypted password file is created using Teamcenter Environment Manager. See *Configuring Teamcenter mail* in the *Application Administration* guide for instructions on creating a new password.

-connSecurity

Specifies the type of secure connection to establish when connecting to the mail server.

None	(The default value.) No security connection protocol is used when connecting with the mail server.
SSL/TLS	Establish an SSL/TLS connection when connecting to the mail server.
STARTTLS	Establish an initial insecure connection and later convert to a TLS connection with the mail server.

-ssltlsProtocol

Specifies the protocol and version to use when connecting with the mail server.

TLSv1	(The default value.) Transport Layer Security version 1.
TLSv1_1	Transport Layer Security version 1.1.
TLSv1_2	Transport Layer Security version 1.2.
SSLv2	Secure Sockets Layer version 2.
SSLv23	Secure Sockets Layer version 2.3.
SSLv3	Secure Sockets Layer version 3.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- When defining the formatting of the attachment list, each line in the list must consist of an attachment file name and its file format.

Assume that the attachment list file is called **attachment_list.txt**, and the list's content is:

```
C:\temp\BAR_RED.JPG=B
C:\temp\test.txt=T
```

This results in two files being included as attachments, one binary file named **BAR_RED.JPG** and one text file named **test.txt**. Both files are located in the **C:\temp** directory.

```
tc_mail_smtp -subject="my test"
-to=person1@company1.com -to=person2@company2.com
-cc=person3@company3.com
-bcc=person4@company4.com
-server=mail_server_1
-body=C:\temp\test.txt -attachments=C:\temp\attachment_list.txt
-user=person5
```

- Connect with an email server requiring authentication.

```
tc_mail_smtp -subject="Secure Example"
-to=person1@company1.com
-server=secure_mail_server:25
-auth=true -authId=email-name
-pf="C:\Tc\TcSecurity\passwd.pf" -connSecurity=SSL/TLS

-sslTlsProtocol=SSLv23
-body=C:\email\body.txt
-charset="ISO-8859-1"
```

uih_to_xml

Converts existing UIH files to XML files that serve text and error messages.

SYNTAX

uih_to_xml **-u**=*user-id* {**-p**=*password* | **-pf**=*password-file*} [**-g**=*group*]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

file1 .uih file2 .uih...

Lists UIH files to be converted to XML. If no file is specified, the current directories and subdirectories are searched for UIH files, which are then translated to XML.

-d= *my / directory*

Parses XML files in a given directory and related subdirectories.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

There are two basic examples for this utility.

- The following example traverses the current directory and all subdirectories searching for UIH files and translates them to XML files:

```
uih_to_xml
```

- The following example translates only the specified UIH files into their corresponding XML files:

```
uih_to_xml    ss_errors.uih    ae_errors.uih
```

ums_import_unit_definitions

Creates, updates, and configures the display of Teamcenter units of measure.

ums_import_unit_definitions runs automatically during Teamcenter installation and can subsequently be run by a Teamcenter administrator to manage units of measure. Siemens Digital Industries Software strongly recommends backing up any current unit definitions file before modifying the file.

Units of measure are created and managed based on unit of measure definitions specified in a comma-delimited .csv file. The initial unit of measure definitions file supplied with Teamcenter is `<TC_DATA>\unit_definitions.csv`. See Working with the unit definitions file for a description of this file.

After running **ums_import_unit_definitions**, ensure users log out of Teamcenter start new sessions to use the updated units of measure.

Use the **ums_mapping** utility to map and add legacy units of measure to the Unit Management System.

SYNTAX

```
ums_import_unit_definitions
[-u=user-id {-p=password | -pf=password-file} -g=group]
-file=unit_def_file
-applications={app_name1,app_name_2,... | all}
[-allowOverwrite]
[[-createUOM] [-updateUOM]] | [-hideUOM] | [-showUOM]
[-dryRun]
[-h]
```

ARGUMENTS

-u

Specifies the user ID. The user must have Teamcenter administration privileges and be a member of a group with **DBA** privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. This group must have **DBA** privileges.

If used without a value, the user's default group is assumed.

-file

The comma-delimited .csv file (*unit_def_file*) containing unit of measure definitions.

-allowOverwrite

Optional. Overwrite existing units of measure based on the definitions in *unit_def_file*.

-applications

Restrict updates to one or more applications (as named in *unit_def_file*) containing unit of measure definitions. *app_name* is the string following **Application:** in *unit_def_file*. For example, set *app_name* to **NX** for the column **Application:NX**.

Separate multiple application names with commas. Set **-applications** to **all** to process all applications defined in *unit_def_file*.

-createUOM

Optional. Create the units of measure defined in *unit_def_file*. If **-allowOverwrite** is not specified, only new units of measure are created. If **-allowOverwrite** is specified, existing units of measure are overwritten based on the definitions in *unit_def_file* as well. **-createUOM** cannot be used with **-hideUOM** or **-showUOM**.

If **-createUOM**, **-updateUOM**, **-hideUOM**, or **-showUOM** are not specified, **-createUOM** is assumed.

-updateUOM

Optional. Update existing units of measure with the definitions in *unit_def_file*. Units of measure defined in *unit_def_file* that do not currently exist will not be created. **-updateUOM** cannot be used with **-hideUOM** or **-showUOM**.

-hideUOM

Optional. Suppress existing units of measure defined in *unit_def_file* from appearing in Teamcenter and application lists of values. These units of measure are not removed from Teamcenter. **-hideUOM** cannot be used with **-createUOM**, **-updateUOM**, or **-showUOM**.

-showUOM

Optional. Show units of measure defined in *unit_def_file* that have previously been hidden using **-hideUOM**. **-showUOM** cannot be used with **-createUOM**, **-updateUOM**, or **-hideUOM**.

-dryRun

Optional. Run **ums_import_unit_definitions** with the specified parameters without actually making changes to the units of measure. The results of running the utility are displayed in the command window, but no units of measure changes are made.

Siemens Digital Industries Software strongly recommends using **-dryRun** to validate changes before actually creating or updating your UMS definitions.

-h

Displays help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment.

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

None.

EXAMPLES

- Create units of measure for all units of measure defined in a .csv file. All existing units of measure are updated with any updated definitions.

```
ums_import_unit_definitions -file=c:\temp\unit_defs.csv
                             -applications=all
```

- For units of measure that are members of the Base application, update existing units of measure with changed definitions.

```
ums_import_unit_definitions -file=c:\temp\unit_defs.csv
                             -applications=Base -allowOverwrite -updateUOM
```

- Suppress units of measure in the Base application from appearing in lists of values.

```
ums_import_unit_definitions -file=c:\temp\unit_defs.csv
                             -applications=Base -hideUOM
```

- Display units of measure in the Base application in lists of values.

```
ums_import_unit_definitions -file=unit_definitions.csv
                             -applications=Base -showUOM
```


Document management

dm_attrexch_xml_validate

Validates the attribute exchange mapping XML configuration format only. If the XML file is valid, it converts the special XML characters (&, <, >, ' , ") to XML encodings and breaks up to two thousand chunks separated by a comma (,) and outputs it to a **ConvertAttrExchXML.txt** file. This converted data can be used for the **bulk attribute import** utility to update the **fnd0InstanceAttrExMappings** mapping attribute on the selected dataset.

Note:

This utility is not an XML validator. It only validates attribute exchange mapping XML configuration format. Make sure the input XML file is a valid XML file.

SYNTAX

```
-inputfile=input-xml-file -logpath=current directory
-bulkfile=bulk-xml-file
-convfile=convert-file
-replacertext=replace-text
```

ARGUMENTS

-inputfile

Specifies the full path of the input XML file that contains the attribute exchange mapping XML configuration.

-logpath

Specifies the log file and output file path (optional).

-bulkfile=

Specifies the full path of the input file that is generated from **attribute_export**, for example, **BulkAttrOut_1.xml**.

-convfile=

Specifies the full path of the input XML file that contains the attribute exchange mapping XML, for example, **ConvertAttrExchXML.txt**.

-replacertext=

Specifies the text that is to be replaced in the bulk file with the text from the converted file, for example, **ReplaceMe**.

-h

Displays help for this utility.

ENVIRONMENT

None.

FILES

None.

RESTRICTIONS

To be run by a DBA user and run in the Teamcenter command prompt.

EXAMPLES

- To display help about the **dm_attrexch_xml_validate** utility, enter the following command on a single line:

```
dm_attrexch_xml_validate -h
```

- To find out if the **C:\data\attrexch.xml** file is valid attribute exchange configuration XML data, generate the **ConvertAttrExchXML.txt** file by entering the following command on a single line:

```
dm_attrexch_xml_validate -inputfile=C:\data\attrexch.xml  
  
-logpath=C:\data
```

- To replace text (-replacetext) in the input **C:\Data\BulkAttrOut_1.xml** file with the data from the **C:\Data\ConvertAttrExchXML.txt** input file, generate the **RBulkAttrOut_1.xml** file by entering the following command on a single line:

```
dm_attrexch_xml_validate -bulkfile=C:\Data\BulkAttrOut_1.xml  
-convfile=C:\Data\ConvertAttrExchXML.txt  
  
-replacetext=ReplaceMe -logpath=C:\Data
```




14. Teamcenter Integration for NX utilities

export_attr_mappings

Exports attribute mappings from the Teamcenter database to a text file.

SYNTAX

```
export_attr_mappings -file=text-file [-test]
[-u=user-id {-p=password | -pf=password-file} -g=group]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the name of the file to be written to.

-test

Exports the test mappings rather than the actual mappings.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

import_attr_mappings

Imports attribute mappings into the Teamcenter database. Siemens Digital Industries Software recommends that you initially import the mappings using the **-test** argument and verify the accuracy of the mappings before making them generally available.

SYNTAX

```
import_attr_mappings -file= text-file [-test] [-dryrun]
[-u=user-id {-p=password | -pf=password-file} -g=group] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file= text-file

Specifies the name of the input file.

-test

Imports the file without overwriting the existing mapping file.

-dryrun

Parses the file but does not save the mappings.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

tess_server

Starts the tessellation server which translates **UGMASTER/UGALTREP** datasets to JT datasets.

SYNTAX

```
tess_server {-s | -c} [-h] [-u=user-id {-p=password | -pf=password-file} -g=group]
{-s | -c} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-s

Starts the tessellation server.

-c

Stops the tessellation server.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Enter the following command to start the tessellation server:

```
tess_server -s
```

- Enter the following command to stop the tessellation server:

```
tess_server -c
```

- Enter the following command to display help for the **tess_server** utility:

```
tess_server -h
```




15. Integration utilities

Linked Data Framework utilities

create_botype_reader

Supports adding new resources and datasets to new Linked Data Framework domains.

SYNTAX

```
create_botype_reader
[-u=user-id {-p=password | -pf=password-file} -g=group]
[-mode=map_schema|resource]
[-type=type-name]
[-protocol=protocol-name]
[-versions=protocol-version]
[-domain=domain-name]
[-file=file-path]
[-f=create|validate|list]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode=map_schema|resource

Specifies whether to create a mapping file or a new resource.

-type

Specifies the name of dataset or resource.

-protocol

Specifies the Linked Data Framework protocol.

-versions

Specifies the version of the Linked Data Framework protocol.

-domain

Specifies the domain to which dataset or resource is being associated.

-f

Specifies whether to create, validate, or list the semantic data. You can use the following values:

- **create:** Creates a dataset that contains mapping information. If the dataset exists, the named reference of the latest data is replaced with the new dataset named reference after the validation is done.
- **validate:** Checks if the Teamcenter type follows the same hierarchy as the extended semantic types.
- **list:** Lists information about each semantic and Teamcenter type in the file specified by the **-filename** argument. The listed information consists of Teamcenter attributes, Teamcenter classes and its attributes for the given semantic type, relations that can be associated with the semantic type, and the OSLC mapping for the relation for consumer and provider use cases.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Adding a new class:

```
create_botype_reader -u=Tc-admin-user -p=password -g=group -mode=resource  
-type=Cus0RMResourceAttrHelper -protocol=oslc -versions=2.0  
  
-domain=rm
```

- Create a new mapping schema:

```
create_botype_reader -u=Tc-admin-user -p=password -g=group  
-mode=map_schema  
-domain=rm -f=create -file=d:\RM_Attr_Mapping_Dataset.xml
```


create_ldf_objecttypes_dataset

Creates a dataset that contains mapping information between Teamcenter objects and objects in the external application.

SYNTAX

```
create_ldf_objecttypes_dataset  
[-u=user-id {-p=password} -g=group]  
[-file=file-path]  
[-f=create]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the file path of the XML file containing mapping between Teamcenter objects and objects in the external application..

-f

Specifies that the dataset should be created.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
create_ldf_objecttypes_dataset -u=Tc-admin-user -p=password -g=group  
-f=create  
-file=D:\tc_eng\LDF\IntegrationTesting\utility\RV1_LDF  
_objectTypes_mapping_file_name.xml
```

generate_remote_links_report_for_update

Generates a report that shows the relation types to be updated.

SYNTAX

```
generate_remote_links_report_for_update  
[-u=user-id {-p=password} -g=group]  
[-filepath=file-path]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-filepath

Specifies the file path of the dataset where the report is generated.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Authenticate external systems before running this utility.

EXAMPLES

```
generate_remote_links_report_for_update -u=Tc-admin-user -p=password
-g=group
-filepath= D:\generate_remote_links_report_for_updatereport.txt
```

The file must be empty before running the utility. The sample report contains the following information:

```
ServiceProvider:PrimaryObject:PrimaryObjectType:PrimaryObjectUID
:ExternalObjectID:LinkType:LinkObjectUID:RelationID:
AvailableRelationsForUpdate
```

For example:

```
TcLDFConnectorProject01:ECR Object:ChangeRequestRevision:AaUdNQZJIn05DC
:TcLDFConnectorProject01-549:Issue:AqedNQZJIn05DC:Relation Type ID:
TYSdNMAvIn05DC:Lcm0AffectsPlanItem=Affects Plan Item,
Lcm0ImplementedBy=Implemented By
```

```
TcLDFConnectorProject01:ECR Object:ChangeRequestRevision:Q0ZdNQZJIn05DC
:TcLDFConnectorProject01-550:Issue:QWcdNQZJIn05DC:Relation Type ID:
TYSdNMAvIn05DC:Lcm0AffectsPlanItem=Affects Plan Item,
Lcm0ImplementedBy=Implemented By
```

maintain_consumer_key

Supports creating or deleting OAuth consumer keys.

SYNTAX

```
maintain_consumer_key
[-u=user-id {-p=password | -pf=password-file} -g=group]
[-mode=add|delete]
[-consumer_name=Linked-Data-Framework-consumer-name]
[-consumer_secret=secret-code]
[-consumer_key=existing-consumer-key]
[-isTrusted=Y|N]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-mode

Specifies whether to add or delete an OAuth consumer key.

-consumer_name

Specifies the name of Linked Data Framework consumer.

-consumer_secret

Specifies an OAuth key. This argument is used only when the argument **-mode=add**.

-consumer_key

Specifies the existing OAuth consumer key. This argument is used only when the argument **-mode=delete**.

-isTrusted

Specifies if the consumer is trusted.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Adding an OAuth consumer key:

```
maintain_consumer_key -u=username -p=password -g=dba -mode=add
-consumer_name=RTC -consumer_secret=111 -isTrusted=Y
```

- Deleting an OAuth consumer key:

```
maintain_consumer_key -u=username -p=password -g=dba -mode=delete
-consumer_name=RTC -consumer_secret=111

-consumer_key=12789259677583223 -isTrusted=Y
```

maintain_ldf_semantics

Adds the relation type between a Teamcenter object type and the object type in the external application to the database. The relation type is specified in an XML file.

SYNTAX

```
maintain_ldf_semantics  
[-u=user-id {-p=password | -pf=password-file} -g=group]  
[-f=create|delete]  
[-file=file-path]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies whether to create or delete semantic data.

-file

Specifies the path of the semantic data file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

```
maintain_ldf_semantics -u=Tc-admin-user -p=password -g=group -f=create
-file=D:\tc_eng\LDF\IntegrationTesting\utility\semantic_data.xml
```

The sample semantic file contains information as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SemanticList>
  <SemanticKey>
    <ContextPrimaryObjectType>ItemRevision</ContextPrimaryObjectType>
    <RemoteObjectType>Defect</RemoteObjectType>
    <SemanticType>Lcm0AffectsPlanItem</SemanticType>
  </SemanticKey>
</SemanticList>
```


migrate_relations_to_ldf

Updates the relation types **IMAN_reference** and **IMAN_specification** that are used for associating Linked Data Framework links as a secondary object to the **Lis0Reference** and **Lis0Specification**, respectively. You must have system administration privileges to run this utility.

SYNTAX

```
migrate_relations_to_ldf
[-u=user-id {-p=password | -pf=password-file} -g=group]
[-report=path-of-the-log-file]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-report

Specifies the file path where the log file is created.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Authenticate external systems before running this utility.

EXAMPLES

```
migrate_relations_to_ldf -u=Tc-admin-user -p=password -g=group  
-report= D:\generate_remote_links_report_for_updatereport.txt
```

update_remote_link_relations

Updates the relation type between a Teamcenter object type and the object type in the external system. Before using this utility, you must update the file generated by the **generate_remote_links_report_for_update** utility.

SYNTAX

```
update_remote_link_relations
[-u=user-id {-p=password} -g=group]
[-filepath=file-path]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-filepath

Specifies the file path of the dataset that contains information about the new relation type to be associated with the primary object and the link object.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Authenticate external systems before running this utility.

EXAMPLES

```
update_remote_link_relations -u=Tc-admin-user -p=password -g=group
-filepath= D:\generate_remote_links_report_for_updatereport.txt
```

You must first update the file generated by the **generate_remote_links_report_for_update** utility with the new relation before running this utility. The format of the input file is as follows:

```
ServiceProvider:PrimaryObject:PrimaryObjectType:PrimaryObjectUID
:ExternalObjectID:LinkType:LinkObjectUID:RelationID:RelationForUpdate
```

For example:

```
TcLDFConnectorProject01:ECR Object:ChangeRequestRevision:AaUdNQZJIn05DC
:TcLDFConnectorProject01-549:Issue:AqedNQZJIn05DC
:Relation Type ID:TYSdNMAvIn05DC:Lcm0AffectsPlanItem
TcLDFConnectorProject01:ECR Object:ChangeRequestRevision:Q0ZdNQZJIn05DC
:TcLDFConnectorProject01-550:Issue:QWcdNQZJIn05DC
:Relation Type ID:TYSdNMAvIn05DC:Lcm0ImplementedBy
```

Teamcenter Community Collaboration

tcc_context_upload

Uploads fully or partially configured assemblies to Teamcenter Community Collaboration after downloading them from Teamcenter. This utility does the following:

1. Sets up the staging directory to store the temporary files.
2. Executes a search in RDV setup and downloads the resulting data into a flat file (AJT or PLM XML) by calling the **rdv_context_download** utility.
3. Calls the **asciitajt** utility to convert the AJT file downloaded in Step 2 into the JT format.
4. Uploads the JT assembly files into Teamcenter Community Collaboration.
5. Deletes the staging directory.

SYNTAX

```
tcc_context_upload [-u=user-id {-p=password | -pf=password-file} -g=group]
[-item_id=item-ID] [-rev_id=revision-ID]
[-variant_rule_name=variant-rule-name] [-revision_rule_name=revision-rule-name]
[-engg_change_id=engineering-change-ID]
[-sco_name=structure-context-object name]
[-folder_name=folder-name] [-process_name=process_name] [-zone_name=zone-name] [-
zone_type=BOX | PLANE] [-title=JT-file-base-name] [-stage=staging-area] [-d] [-verbose][-
tccuser=TeamcenterCommunity-user-name] [-tccpass=TeamcenterCommunity-password] [-tccanon] [-
tccreds=TeamcenterCommunity-credential-file] [-tccdestination=url] [-keep] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-sco_name

Specifies the name of the structure context object.

-item_id

Specifies the item ID.

-rev_id

Specifies the revision ID.

-key

Specifies the key of the item. Use the following format:

```
[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]
```

To find the key of an object, use the **get_key_string** utility.

To find the key of an object, use the **get_key_string** utility.

-engg_change_id

Specifies the engineering change item ID. The utility configures an RDV context based on change attachments and the latest engineering change revision.

-process_name

Specifies the name of the Teamcenter workflow processes that have not yet completed.

-zone_name

Specifies the name of the zone. When both the **-zone_name** and **-zone_type** arguments are specified, the utility performs an appearance or QPL search according to the preference settings.

-zone_type

Specifies the type of the zone, which must be either **BOX** or **PLANE**. The **-zone_name** argument must be used in conjunction with the **-zone_type** argument.

-folder_name

Configures a context based on attachments of the *folder/envelope/engineering-change-revision-name*.

The attachments include the following:

- One product item revision
- One or more component item IDs
- Optional revision rule, overwrites the **-r** argument
- Optional variant rule, overwrites the **-v** argument

Note:

Search results are affected by the following Teamcenter preferences:

- **TC_config_rule_name**
- **WebDesignContextDefaultSearchDistance**
- **PortalDesignContextMaxMatchingObjects**
- **PortalDesignContextMaxMatchingBOMLines**

For more information, see the *Teamcenter Environment Variables*.

-variant_rule_name

Specifies the name of the variant rule.

-revision_rule_name

Specifies the revision rule, which defaults according to the **TC_config_rule_name** preference.

-title

Specifies the base name of the JT file. If not specified, it defaults to **-l**.

-stage

Specifies the staging area (for example, **/tmp**). If not specified, it defaults to **\$TC_TMP_DIR**.

-d

Turns on debugging for the **Create bookmark (rdv_context_download)** utility.

-verbose

Turns on debugging for this utility.

-tccuser

Specifies the Teamcenter Community Collaboration user name if it is different than the user name in the **-u** argument.

-tccpass

Specifies the Teamcenter Community Collaboration password if it is different than the password in the **-p** argument.

-tccanon

Sets the Teamcenter Community Collaboration login to anonymous, assuming the destination permits anonymous access.

-tcccreds

Specifies the Teamcenter Community Collaboration externally created credential file for logging in.

-tccdestination

Specifies the URL for Teamcenter Community Collaboration.

-keep

Do not delete the staging area after uploading the JT assembly files.

-h

Displays help for this utility.

ENVIRONMENT

- The **TcCUploaderS.jar** file must be present in **CLASSPATH**.
- The Teamcenter environment must be set to run this utility.
- As mentioned in the *Dependencies* section below, the user must have a Teamcenter Community Collaboration login and a Teamcenter Community Collaboration environment set up.
- If the **IMAN_QPL_PROX_FILTER_INCL_COMPS** environment variable is set, proximity searches not only return nearby parts for an instance, but also for all of its children and grandchildren.
- If the **IMAN_RDV_VALID_OVERLAYS_ONLY** environment variable is set, background searches (the **-c** or **-n** argument) return only background component instances overlaying valid combinations of variants. This may cause long processing times.
- If the environment variable **RDVContextDownloadDebug=1** is set, the Teamcenter **syslog** file contains information about the **rdv_context_download** utility.
- If the environment variable **RDV_debug=Init+QPL+SearchCriteria+Variants** is set, the Teamcenter **syslog** file contains additional RDV debugging information.

DEPENDENCIES

- This utility depends on the **TcCUploaderS.jar** file. Before using this utility, you must get this file from the Teamcenter Community Collaboration kit and place it in the **CLASSPATH** on each client host.
- You must have access to the Teamcenter Community Collaboration environment to run this utility.

RESTRICTIONS

- Zone searches (-z) require an NX-based QPL search index with zones or zone filters explicitly created in a product structure for appearance caches.
- Context searches (-c and -n) require a QPL search index.

The attachments include the following:

- One product item revision
- One or more component item IDs
- Optional revision rule, overwrites the -r argument
- Optional variant rule, overwrites the -v argument

Note:

Search results are affected by the following Teamcenter preferences:

- **TC_config_rule_name**
- **WebDesignContextDefaultSearchDistance**
- **PortalDesignContextMaxMatchingObjects**
- **PortalDesignContextMaxMatchingBOLMLines**

For more information, see the *Teamcenter Environment Variables*.

EXAMPLES

- The following example uploads the antenna assembly to Teamcenter Community Collaboration, overlaying all applicable variants (the RDV search index is not used or needed). It uses the **TC_config_rule_name** user preference to determine the revision rule.

```
$TC_ROOT/bin/tcc_context_upload -item_id TL109375 -rev_id 004
-title Antenna -stage /tmp -tccuser subrata
-tccdestination http://usamseveh001/mydocs
```

- The following example uploads the antenna assembly to Teamcenter Community Collaboration, overlaying all applicable variants (the RDV search index is not used or needed). It enforces a revision configuration using the **Beta or less w/pdi** revision rule.

```
$TC_ROOT/bin/tcc_context_upload -item_id TL109375 -rev_id 004
-title Antenna -revision_rule_name "Beta or less w/pdi" -stage /tmp
-tccuser subrata -tccdestination http://usamseveh001/mydocs
```

- The following example uploads a subset of the RDV00190 product assembly that contains all components in the ENGINE zone to Teamcenter Community Collaboration. It also:
 - Requires an NX ENGINE box zone in the top-level NX part file
 - Requires an NX-based QPL search index
 - Overlays all applicable variants
 - May require a higher value for the **PortalDesignContextMaxMatchingBOLMLines** preference if the ENGINE zone has many components.

```
$TC_ROOT/bin/tcc_context_upload -item_id RDV00190 -rev_id 008
-title EngineCompartment - revision_rule_name "Beta or less w/pdi"
-zone_name ENGINE -stage /tmp -tccuser subrata
-tccdestination http://usamseveh001/mydocs
```

- The following example uploads a subset of the product assembly referenced in the latest revision of the 000042RDV Engineering Change Item, including affected components and their nearby parts within the distance specified in the **WebDesignContextDefaultSearchDistance** preference. Please note:
 - 000042RDV is expected to reference the following:
 - The affected parts or part revisions (for example, 15759576/003-RADIATOR ASM-(W/ A/C CNDSR) 15006864/015-REINF-RAD UPPER INR SUPT LH 15068174/002-SUPPORT_ASM_RADIATOR).
 - The product item revision (for example, RDV00190/008).
 - The revision rule (for example, **Beta or less w/pdi**)
 - The subset contains all components inside the ENGINE zone.
 - It requires a QPL search index.

- It overlays all applicable variants.

```
$TC_ROOT/bin/tcc_context_upload -engg_change_id 000042RDV  
-title RadiatorSupport -stage /tmp -tccuser subrata  
-tccdestination http://usamseveh001/mydocs
```

Exporting Teamcenter data to Intosite

tc_to_intosite

Exports Teamcenter data to Intosite by traversing a designated bill of equipment (BOE) or bill of process (BOP) structure and creating an Excel file with Intosite objects that correspond to the Teamcenter objects.

SYNTAX

```
tc_to_intosite -u=user-id {-p=password | -pf=password-file} [-g=group]
```

```
[-mappingfile=]
[-closuresrule=]
[-objectsafterdate=]
[-cc= -ccuid= -ccsc= -ccscuid= ]
[-sc= -scuid= ]
[-scope= ]
[-intositeobjectstoexport= ]
[-outexcel= ]
[-outputdir= ]
[-tcawurl= -tcewiurl= -tceurl= -tc4racurl= -tcintositeurl= ]
[-inputaddressfile= -outputaddressfile= -outputaddressproperty= ]
[-intositeurl= -intositecompany= -intositeuser= -intositepasswd= -intositeuploadnotify=]
[-loglevel= -log= ]
[-constantKey=constantValue ]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If not specified, the user's default group is assumed.

-mappingfile

XML file that defines mapping between Teamcenter and Intosite objects. If not specified, uses value from preference **TcToIntosite_MappingFile**.

-closuresrule

Closure rule to use to find candidate Teamcenter objects. If not specified, uses value from preference **TcToIntosite_ClosureRule**.

-objectsafterdate

After objects are collected by closure rule, then find Teamcenter objects modified after specified date.

Format: **dd-mmm-yyyy HH:MM** where mmm is abbreviated (example: **07-Jul-2020 14:25**).

-cc

Name of Collaboration Context object that contains the structure to use. Either a Collaboration Context or Structure Context must be specified.

-ccuid

Specifies the Collaboration Context using UID identifier instead of **-cc=<name>**.

-ccsc

In the case where Collaboration Context contains multiple structures, specifies the name of the specific Structure Context to use.

-ccscuid

In the case where Collaboration Context contains multiple structures, specifies the uid of the specific Structure Context to use.

-sc

Name of Structure Context object to use. Either a Collaboration Context or Structure Context must be specified.

-scuid

Specifies the Structure Context using UID identifier instead of **-sc=<name>**.

-scope

Item ID specifying scope to use within the Structure Context. If not specified, the root of the structure is the scope.

-intositeobjectstoexport

Comma delimited list of Intosite object types to export. Default all objects in mapping file.

-outexcel

Name of the excel output file name. If not specified, uses **TcToIntosite**.

-outputdir

Full path of directory where excel file will be created. If not specified, uses current working directory.

-tcawurl

URL of Active Workspace server.

-tcewiurl

URL of Electronic Work Instructions.

-tcepurl

URL of Easy Plan.

-tc4racurl

URL of Teamcenter server.

-tcintositeurl

URL of Intosite.

-inputaddressfile

This filename specifies an address file. This address file contains address information formatted as follows:

<BOMlinePropertyName>:<value>:Application:<value>:ExternalID:<value>:DX:<value>

:DY:<value>;DALT:<value>

-outputaddressfile

This creates an address file based on -mappingfile address info. The format is as follows:

<BOMlinePropertyName>:<value>:Application:<value>:ExternalID:<value>:DX:<value>;DY:<value>;DALT:<value>

-outputaddressproperty

The <BOMLinePropertyName> to use when creating the output address file.

-intositeurl

URL to Intosite site.

-intositeuser

User to log on to Intosite site.

-intositepasswd

Password to log on to Intosite site.

-intositecompany

Company to log on to Intosite site.

-intositeuploadnotify

Email address to notify after Excel file is uploaded to Intosite site.

-loglevel

DEBUG Additional information beyond error messages.

-log

Log file name. If not specified, log information is dumped to the console.

-constantKey=constantValue

Any pair `<constantKey>=<constantValue>` on the command line is an override of the same constant key/value pair in the mapping file.

-h

Displays help for this utility.

Note:

Links to external files are not allowed in the documentation in pdf format. However, there is an example structure and the XML and Excel file used to export it shown in this same topic in the HTML format of the documentation.

tc_to_intosite_invoker

Introduced in Teamcenter 12.4, this is a client side SOA utility application that invokes the server side **tc_to_intosite** ITK application and returns the generated Excel and log files to the client.

A Teamcenter File Client Cache service must be running for file transfers.

To execute this utility, the SOA client application that interacts with the server side **tc_to_intosite** application must be installed. It is a zip file located in the Teamcenter installation **additional_applications** folder: **additional_applications/tc_to_intosite_invoker/tc_to_intosite_invoker.zip**

Note:

A *readme.txt* file in the zip file describes install and configuration information.

SYNTAX

```
tc_to_intosite_invoker -u=user-id {-p=password | -pf=password-file} [-createPf=password-file] [-g=group]
```

```
-host=
-servermode=
[-mappingfile=]
[-closurerule=]
[-objectsafterdate=]
[-cc= -ccuid= -ccsc= -ccscuid= ]
[-sc= -scuid= ]
[-scope= ]
[-intositeobjectstoexport= ]
[-outexcel= ]
[-outputdir= ]
[-tcawurl= -tcewiurl= -tceurl= -tc4racurl= -tcintositeurl= ]
[-inputaddressfile= -outputaddressfile= -outputaddressproperty= ]
[-intositeurl= -intositecompany= -intositeuser= -intositepasswd= -intositeuploadnotify=]
[-loglevel= -log= ]
[-constantKey=constantValue ]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

-p

Specifies the password. This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user. If not specified, the user's default group is assumed.

-createPf

Name of the file where the encrypted password is to be written. The directory, if specified, must exist. This argument must be used in conjunction with the **-p** option—that is what gets encrypted.

Usage: **tctointositeinvoker -p=<passwd> -createPf=<fileName>**. The created file can be used as the argument to **-pf=<file>**.

-host

<url or iiop> [required] specifies server connection information.

Four-tier example: "http://server:port/tc"

Two-tier example: "corbaloc:iiop:localhost:7777/localhost7777"

Note:

Note the use of quotes.

-servermode

< 2 | 4> 2 = tc_to_intosite_invoker in 2-tier or 4 = tc_to_intosite_invoker in 4-tier

-mappingfile

XML file that defines mapping between Teamcenter and Intosite objects. If not specified, uses value from preference **TcToIntosite_MappingFile**.

-mappingfile= <absolute local path/mapping_file.xml> uploads file to server to use.

or

-mappingfile= <absolute server path/mapping_file.xml> uses the server-based file.

-cluserule

Closure rule to use to find candidate Teamcenter objects. If not specified, uses value from preference **TcToIntosite_ClosureRule**.

-objectsafterdate

After objects are collected by closure rule, then find Teamcenter objects modified after specified date.

Format: **dd-mmm-yyyy HH:MM** where mmm is abbreviated (example: **07-Jul-2020 14:25**).

-cc

Name of Collaboration Context object that contains the structure to use. Either a Collaboration Context or Structure Context must be specified.

-ccuid

Specifies the Collaboration Context using UID identifier instead of `-cc=<name>`.

-ccsc

In the case where Collaboration Context contains multiple structures, specifies the name of the specific Structure Context to use.

-ccscuid

In the case where Collaboration Context contains multiple structures, specifies the uid of the specific Structure Context to use.

-sc

Name of Structure Context object to use. Either a Collaboration Context or Structure Context must be specified.

-scuid

Specifies the Structure Context using UID identifier instead of `-sc=<name>`.

-scope

Item ID specifying scope to use within the Structure Context. If not specified, the root of the structure is the scope.

-intositeobjectstoexport

Comma delimited list of Intosite object types to export. Default all objects in mapping file.

-outexcel

Name of the excel output file name. If not specified, uses **TcToIntosite**.

-outputdir

The absolute local path to download the excel file. If not specified, uses current working directory.

-tcawurl

URL of Active Workspace server.

-tcawiurl

URL of Electronic Work Instructions.

-tceurl

URL of Easy Plan.

-tc4racurl

URL of Teamcenter server.

-tcintositeurl

URL of Intosite.

-inputaddressfile

This file name specifies an address file. This address file contains address information formatted as follows:

```
<BOMLinePropertyName>:<value>:Application:<value>:ExternalID:<value>:DX:<value>
:DY:<value>:DALT:<value>
```

-inputaddressfile= <absolute local path/addressfile.txt> uploads file to server to use.

or

-inputaddressfile= <absolute server path/addressfile.txt> uses the server-based file.

-outputaddressfile

This creates an address file based on the -mappingfile address information. The format is as follows:

```
<BOMLinePropertyName>:<value>:Application:<value>:ExternalID:<value>:DX:<value>:DY:<value>:
DALT:<value>
```

<absolute local path/addressfile.txt> generates the address file and download file to local path.

-outputaddressproperty

The **<BOMLinePropertyName>** to use when creating the output address file.

-intositeurl

URL to Intosite site.

-intositeuser

User to log on to Intosite site.

-intositepasswd

Password to log on to Intosite site.

-intositecompany

Company to log on to Intosite site.

-intositeuploadnotify

Email address to notify after Excel file is uploaded to Intosite site.

-loglevel

DEBUG Additional information beyond error messages.

-log

Creates a log file and downloads the log file to this path **<absolute local path/logfile.txt>**. If not specified, log information is dumped to the console.

-constantKey=constantValue

Any pair *<constantKey>=<constantValue>* on the command line is an override of the same constant key/value pair in the mapping file.

-h

Displays help for this utility.

Use the Teamcenter to Intosite wizard

This Java client wizard assists you in collecting program arguments for the **tc_to_intosite** and **tc_to_intosite_invoker** utilities. The wizard can then execute either utility or you can execute the utilities later using the *.bat* file created by the wizard.

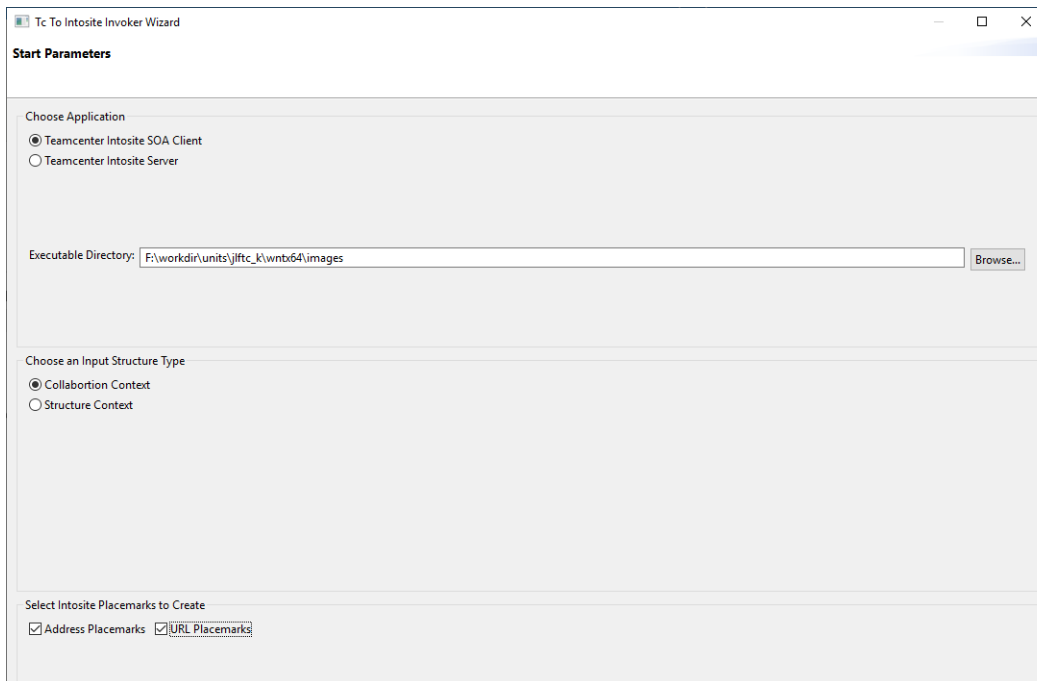
You need to install this wizard, so you can execute it. It is a zip file located in the Teamcenter installation **additional_applications** folder: **additional_applications/tctointositegui/tctointositegui.zip**

Note:

A *readme.txt* file in the zip file describes install and configuration information.

1. Run this command: **java -jar tctointositegui.jar**

The **Tc to Intosite Invoker Wizard** appears, showing the **Start Parameters** page.



Teamcenter Intosite SOA Client and **Teamcenter Intosite Server** options:

- The **Teamcenter Intosite SOA Client** option will execute the **tc_to_intosite_invoker** utility which runs on the client.
- The **Teamcenter Intosite Server** option will execute the **tc_to_intosite** utility which runs on the server.

Collaboration Context and **Structure Context** options:

- Collaboration Context - Used to save groups of data in a specific configuration. It is a Teamcenter object that holds a collection of data contained in structure and configuration contexts.
- Structure Context - Single collection of data.

Note:

The **tc_to_intosite** and **tc_to_intosite_invoker** utilities will not work if a Context is not selected.

Address Placemarks and **URL Placemarks** options:

- Address Placemarks – Pointer to predefined locations within a facility or location.
- URL Placemarks – Pointer to an internet website location.

For more information on Placemarks, see the **User assistance** section in the Intosite documentation.

2. Choose your options in the **Start Parameters** page and click **Next**.

Note:

The options you choose in the **Start Parameters** page determine what pages will display later in the wizard.

The **Host and Credential Parameters** page that appears next contains the information used by the **tc_to_intosite_invoker** utility to communicate with the Teamcenter server.

Tc To Intosite Invoker Wizard

Host and Credential Parameters

Input Host Information

Choose a Protocol

☒ HTTP ☐ HTTPS ☐ IIOP ☐ TCCS

Protocol Information

Host: localhost

Port: 7777

Application: localhostserver

Input Credentials

User: testuser

☒ Password ☐ Password File

Group:

Browse...

3. Select input host information for your environment.
 - HTTP – For Teamcenter 4-tier environment
 - HTTPS – For Teamcenter 4-tier environment
 - IIOP – For Teamcenter 2-tier environment
 - TCCS – Teamcenter client communication system
4. Type your protocol and input credentials information.
5. When done, click **Next** or **Finish**.
6. On the **File Parameters** page, type the Excel, mapping, and log files information, and click **Next** or **Finish**.

Tc To Intosite Invoker Wizard

File Parameters

Input Excel File Information

Excel File Name: TcToIntosite Browse...

Output Directory Name: c:\temp Browse...

Select Mapping File Option

☒ Mapping File Preference: TcToIntosite_MappingFile

☐ Local Mapping File Name: F:\workdir\units\jfftc_klout\ut\data\tc_intosite_mapping.xml Browse...

Input Log File Information

Full Path Log File Name: c:\temp\ins0data\tclog.txt Browse...

Log Level: DEBUG

Note:

Based on what you selected on the **Start Parameters** page in step 1, the next page displayed (**Collaboration Context Parameters**) will show either the **Select Collaboration Context** section or the **Select Structure Context of Collaboration Contest** section, but not both.

7. Type your structure name or UID.

Tc To Intosite Invoker Wizard

Collaboration Context Parameters

Select Collaboration Context

☒ Name: tcToIntositeAT_CC_1583781534_2_27112

☐ UID:

Select Structure Context of Collaboration Context

☒ Name: BOEContext

☐ UID:

Additional Structure Information

Scope Item Id:

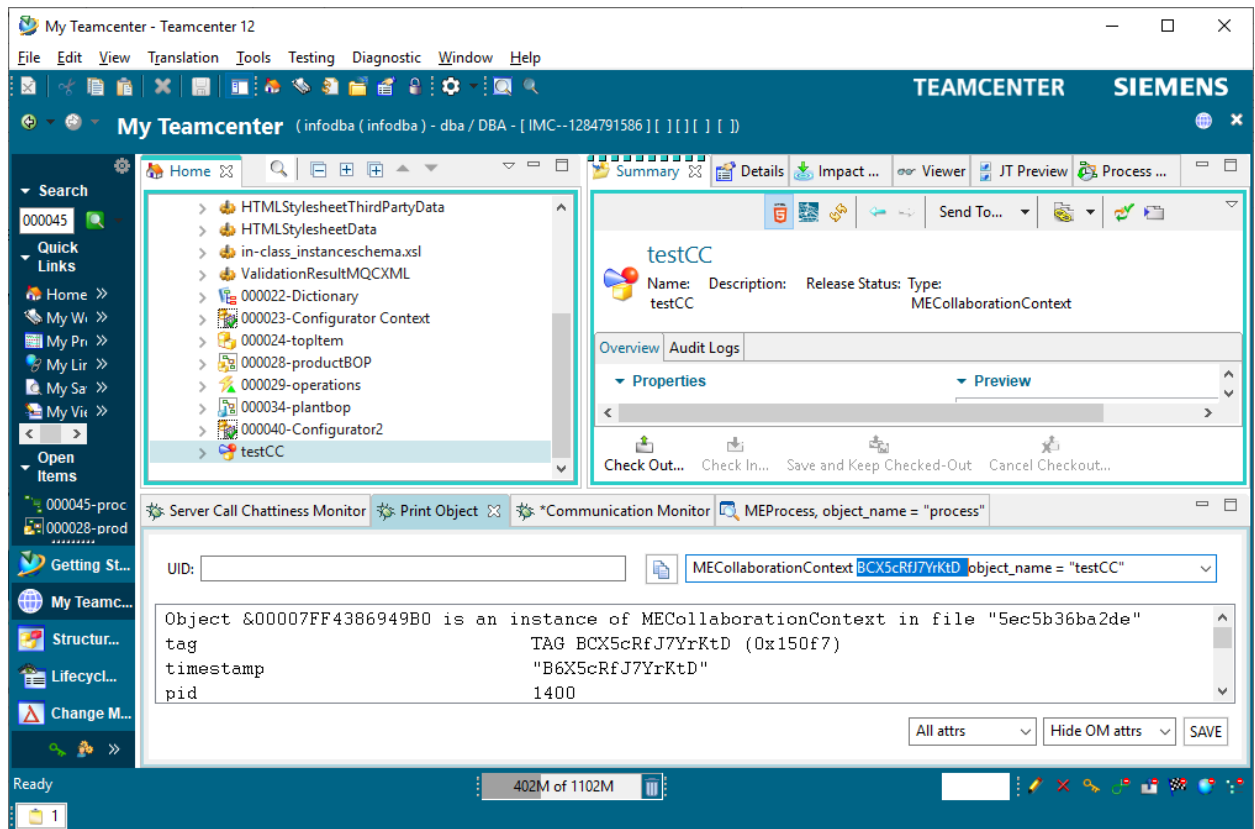
Closure Rule: TcToIntosite_ClosureRule

☐ Collect Objects After Date

March 2020						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
23	24	25	26	27	28	29
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

9:42:00 AM

- The Collaboration Context or Structure Context **Name** must be unique.
- How to find the UID in the Teamcenter rich client.



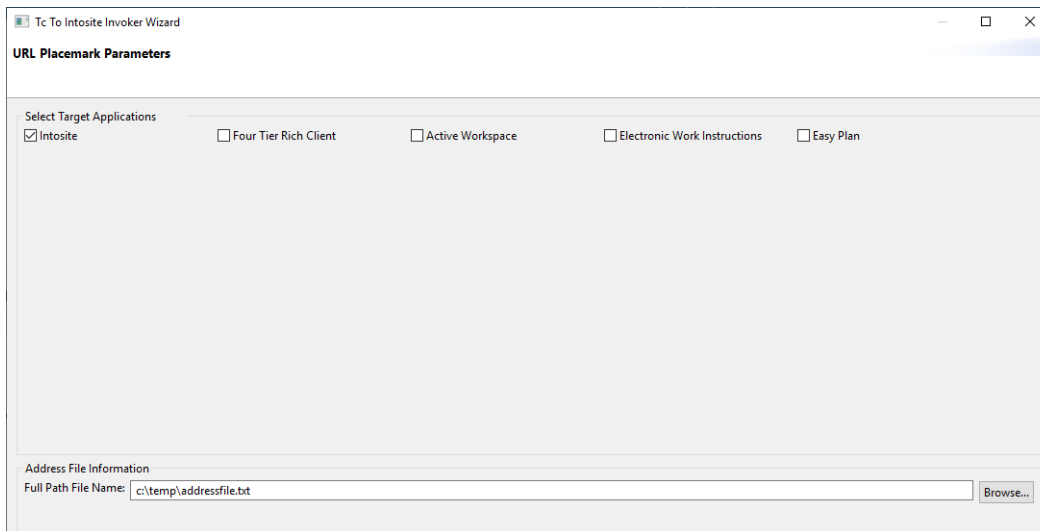
- Scope – Item ID specifying scope to use within the Structure Context. If not specified, the root of the structure is the scope.
8. If needed, enter any **Additional Structure Information**.
 9. When done, click **Next** or **Finish**.
 10. On the **Address Placemark Parameters** page, type the property name and full path to the address file.

**Note:**

The property name must be an internal BOM Line property name.

11. When done, click **Next** or **Finish**.

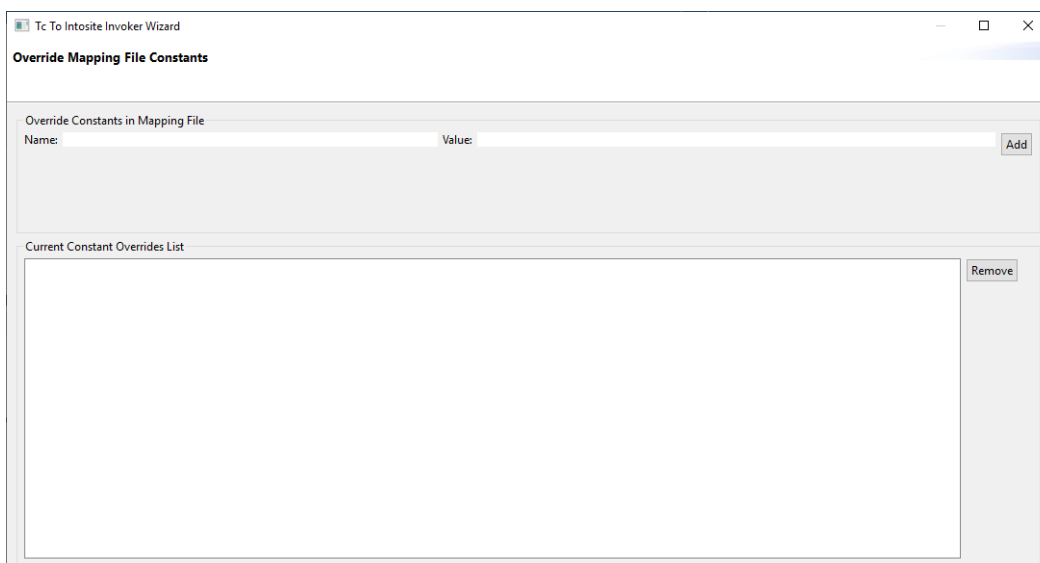
12. On the **URL Placemark Parameters** page, select the target applications.



Note:

Select Intosite if you need file references, studies, or Process Simulate.

13. Click the **Browse** button to locate and select the address file, and click **Next** or **Finish**.
14. On the **Override Mapping File Constants** page, type any override constants you need and click **Add** after each to add it to the list. If you do not want to override any constants, select all of them and click **Remove**.



15. When finished, click **Next** or **Finish**.

16. On the **Upload Excel to Intosite Parameters** page, type all necessary information, and click **Next** or **Finish**.

Tc To Intosite Invoker Wizard

Upload Excel to Intosite Parameters

☐ Upload Excel to Intosite

Input Intosite Credentials

URL:

User:

Password:

Company:

Email Notification:

17. In the **Current Command Line Parameters** page, review the command line listing with all arguments (two examples are shown).

Tc To Intosite Invoker Wizard

Current Command Line Parameters

Current Command Line

```
F:\workdir\units\jlfrc_k\wntx64\images\tc_to_intosite_invoker.exe, -host=http://localhost:7777/localserver, -servermode=4, -u=testuser, -p=jjj, -outputdir="c:\temp", -mappingfile="F:\workdir\units\jlfrc_k\out\rt\data\tc_intosite_mapping.xml", -cc="TcToIntositeAT_CC_1583781534_2_27112", -ccsc="BOEContext", -outexcel="TcToIntosite", -tcintositeurl=http://localhost:7777/localserver, -log="c:\temp\ins0data\tclog.txt", -loglevel=DEBUG, -intositeobjectstoexport=addressPlacemark,urlPlacemark, -inputaddressfile="c:\temp\addressfile.txt", -outputaddressproperty=bl_item_item_id, -outputaddressfile="c:\temp\addressfile.txt"
```

Tc To Intosite Invoker Wizard

Current Command Line Parameters

Current Command Line

```
[F:\workdir\units\jlfrc_k\wntx64\images\tc_to_intosite_invoker.exe, -host=corbaloc:iiop:localhost:7777/localserver, -servermode=2, -u=user, -p=***, -outputdir=""c:\temp"", -mappingfile=""F:\workdir\units\jlfrc_k\out\rt\data\tc_intosite_mapping.xml"", -tcobjectsafterdate=""04-Jun-2020 14:41"", -ccsc=""BOEContext"", -ccuid=BCX5cRfJ7YrKtD, -outexcel=""TcToIntosite"", -tcintositeurl=http://localhost:7777/localserver, -log=""c:\temp\ins0data\tclog.txt"", -loglevel=DEBUG, -intositeobjectstoexport=addressPlacemark,urlPlacemark, -inputaddressfile=""c:\temp\addressfile.txt"", -outputaddressproperty=bl_item_item_id, -outputaddressfile=""c:\temp\addressfile.txt"", SITE=""MY_NEW_SITE""]
```

< Back Next > **Finish** Cancel

18. Click **Save As** to save the command and arguments to a *.bat* file to be executed later or click **Finish** to execute the command immediately.



16. Teamcenter Automotive Edition-GM Overlay utilities

find_released_datasets

Locates the item revisions and item IDs for the **UGMASTER** and **UGALTREP** datasets released after a certain date and that do not have a DirectModel dataset. The output is written to a file, which is a user argument for the utility. If this argument is not provided, output is written to **query_output.txt** in the current location. If no release date is provided, the system date is used.

SYNTAX

```
find_released_datasets [-u=user-id {-p=password | -pf=password-file} -g=group]
-released_date=date [-bypass= TRUE | FALSE]
[-out_file=output-file-name] [include_remote] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-released_date

Date on which the **UGMASTER/UGALTREP** datasets are released to which we are going to create the tessellated datasets. The format is *DD-MMM-YYYY*. This argument is optional. If not specified, system date is used as the released date.

-bypass

Indicates a bypass switch for having bypass privileges. The default value is **FALSE**.

-out_file

Specifies the file to which output is written. This argument is optional. If this is not specified, the file is written to the current location.

-include_remote

If the value given is not null, remote objects are included.

-h

Displays help for this utility.

EXAMPLES

```
find_released_datasets -u=Tc-admin-user -p=password -g=group  
-released_date=22-Mar-2005  
-out_file=c:\temp\find_released_datasets_test.txt
```

get_bvr_structure

Retrieves the assembly structure of a BVR of the specified item revision, and retrieves the child item IDs associated with the assembly structure, then writes these results to the output file along with the IR.

SYNTAX

```
get_bvr_structure [-u=user-id {-p=password | -pf=password-file} -g=group]
[-f=input-file-name | -itemRevisionKeyFile=file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the input file name containing a list of the specified item revisions.

-itemRevisionKeyFile

Specifies the input file name containing the keys of the specified item revisions. The file format is:

```
-key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...
-key =[keyAttr1=keyVal1][keyAttr2=keyVal2]...
-key =[keyAttr1=keyVal1][keyAttr2='keyVal2']...
```

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-h

Displays help for this utility.

Note:

This utility creates a log file in the **/tmp** directory and writes the error messages to this file. This utility also creates the output file in the current working directory and writes the NX component names with the item revision.

EXAMPLES

```
get_bvr_structure -u=Tc-admin-user -p=password -g=group -f=input_file
```

list_ir_with_bvr

Lists the item revisions with BVRs in the database to the **list_ir_with_bvr_XXXXXXX** output file.

SYNTAX

```
list_ir_with_bvr [-u=user-id {-p=password | -pf=password-file} -g=group]
-type=CORP_Part Revision -query=defined-query-name
-create_before=date-time -create_after=date [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-type

Specifies only the item revision type listed in the **list_ir_with_bvr_timeStamp.output** file.

-query

Specifies that the specified query is executed. This query must be defined in the database.

-create_before

Specifies the date to retrieve the list of item revisions created before this date. The format is *dd-mo-yyyy hh:mm*.

-create_after

Specifies the date to retrieve the list of item revisions created after this date. The format is *dd-mo-yyyy hh:mm*.

-h

Displays help for this utility.

migrate_gmo_to_gcn_events

Migrates all subscription events created in the GM Overlay using **CNonIR** project to create the GCN events. Migration is done by modifying attributes such as **event_type**, **attribute_names**, **attribute_values**, **logic_operators**, and **math_operators** on the **ImanSubscription** class.

SYNTAX

```
migrate_gmo_to_gcn_events [-u=user-id {-p=password | -pf=password-file} -g=group]
[-list]
-split=number-of-subscriptions-in-a-file] [-input_file=input-file] [-report=report-file] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-list

Outputs all UIDs of instances of the **ImanSubscription** class in the database.

-split

Limits the number of subscriptions output to each file.

-input_file

Specifies the file containing the UIDs of the subscriptions. Only the valid GM Overlay subscription events specified in the input file are migrated to GCN events.

-report

Writes all messages and errors to the file specified by this argument.

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- To list the UIDs of all the subscriptions in the database in an output file, enter the following command on a single line:

```
migrate_gmo_to_gcn_events -user=test-user -p=test-password -g=dba -list
```

- To output all UIDs of the subscriptions in the database to output files containing a maximum of 100 subscriptions per file, enter the following command on a single line:

```
migrate_gmo_to_gcn_events -user=test-user -p=test-password -g=dba -list  
-split=100
```

- To migrate all subscriptions in a given input file and write all messages and errors to a file specified by the report option, enter the following command on a single line:

```
migrate_gmo_to_gcn_events -user=test-user -p=test-password -g=dba  
-input_file=/tmp/migrate_input.txt -report=/tmp/migrate_report.txt
```

If the report option is not specified, the utility writes to the **migrate_gmo_to_gcn_events.txt** default file.

- To migrate all subscriptions in the database and report all messages and errors to the default report file using the default user login, enter the following command on a single line:

```
migrate_gmo_to_gcn_events
```

gmo_assoc_items_to_project

Converts special GM logic in the object description field to the project level security feature, as follows:

1. Queries all projects in the database.
2. For each project, searches the description field of all item revisions in the database for the following string:

`|project-id |`

If a match is found, the utility associates the corresponding item with the project.

The utility assumes the following:

- The pipe symbol is the delimiter used in the object description field.
- An item may be assigned to two different projects.
- Valid projects exist in the database.

SYNTAX

gmo_assoc_items_to_project [-u=user-id {-p=password | -pf=password-file} -g=group] [-h]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

None.

RESTRICTIONS

None.

EXAMPLES

- To obtain help for this utility, enter the following command on a single line:

```
gmo_assoc_items_to_project -h
```

- To associate all items in the database to appropriate projects based on the object description field, enter the following command on a single line:

```
gmo_assoc_items_to_project -u=Tc-admin-user -p=password -g=group
```

gmo_change_itemid_naming_rule

Modifies the item ID naming rule property from a given value to the customer specified value.

SYNTAX

```
gmo_change_itemid_naming_rule [-u=user-id {-p=password | -pf=password-file}
-g=group] -option=option -prefix=prefix -nr_name=naming-rule
-init_value=initial-value -max_value=maximum-value [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-option

Specifies the option for this utility, that is, **change_naming_rule**.

-prefix

Specifies the prefix of the GM ItemNumeric naming rule.

-nr_name

Specifies the naming rule to change the prefix. You can specify more than one naming rule separated by commas.

-init_value

Specifies the starting value of the counter.

-max_value

Specifies the maximum value of the counter.

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- The following example uses the **GM ItemRule** naming rule to change the prefix:

```
gmo_change_itemid_naming_rule -u=user -p=password -g=group
  -option=change_naming_rule -nr_name="GM ItemRule "
  -prefix=GMO -init_value=00000 -max_value=99999
```

- The following example uses the **GM ItemRule** and **CORP_Tool Rule** naming rules to change the prefix:

```
gmo_change_itemid_naming_rule -u=user -p=password -g=group
  -option=change_naming_rule -nr_name="GM ItemRule,CORP_Tool Rule"
  -prefix=GMO -init_value=00000 -max_value=99999
```

gmo_change_owner

Sets user and group ownership of objects contained in a folder, item, or item revision. All processing procedures are logged to the log file, if specified.

SYNTAX

```
gmo_change_owner [-u=user-id {-p=password | -pf=password-file} -g=group]
-folder=folder-name -item=item-id -rev=revision-id
[-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
[-r] -owner=new-user-id -own_grp=new-group
-log=log-file-name -bypass [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-folder

Specifies the folder name. Either **folder** or **item** and/or **rev** should be supplied.

-item

Specifies the item ID.

-rev

Specifies the revision ID.

-key

Specifies the key of the item to be checked. The **-key** argument can be specified instead of the **-item** argument. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-r

If specified, the ownership of the contents is changed.

-owner

Specifies the user ID to which the ownership is changed. Both **owner** and **own_grp** should be supplied.

-own_grp

Specifies the new group to which the ownership is changed.

-log

Specifies verbose messages are logged to this file.

-bypass

Bypass access checks. This argument is available only to the system administrator.

-h

Displays help for this utility.

gmo_check_comp_names

Truncates the component names greater than 25 characters in length. This program locates the components of an assembly with component names greater than 25 characters long. The component name is the occurrence note of type name **UG NAME**.

SYNTAX

```
gmo_check_comp_names [-u=user-id {-p=password | -pf=password-file} -g=group]
-all | -f=folder-name | -i=item-id
| [-key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
[-r=itemrev-id] [-report | update] [-out=output-file-name]
[-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-all

Runs on all the items in database.

-f

Specifies the name of the folder containing list of items to be checked for component names.

-i

Specifies the item ID to be checked.

-key

Specifies the key of the item to be checked. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-r

Specifies the revision ID to be checked.

-report

Generates the report of items with component names greater than 25 characters.

-update

Truncates all the component names greater than 25 characters.

-out

Specifies the name of the file to which output is written. This argument is optional. Default file name is **gmpdm_comp_name_report.txt**.

-v

Specifies verbose mode.

-h

Displays help for this utility.

RESTRICTIONS

This utility must be run by a user in the **dba** group with the **-update** option.

EXAMPLES

```
gmo_check_comp_names -u=Tc-admin-user -p=password -g=group -all -report -v
```

gmo_clone

Serves as a wrapper over the Teamcenter Integration for NX **ug_clone** utility to provide Teamcenter Automotive Edition–GM Overlay-specific clone import/export functionality from a command shell.

Use this utility to import and export NX data in the GM Overlay environment to ensure the clone conforms with GM Overlay naming conventions.

SYNTAX

```
gmo_clone [-pim=yes] [-u=user-id {-p=password | -pf=password-file} -g=group]
[-corba_ior_file=ior-file] [-http_url=4-tier-server-url]
[-o=operation] [-fam=lose|strip_status|error] [-asse=assembly] [-par=part] [-dir=directory-
name] [-fol=folder] [-default_checki=default-check-in] [-default_checko=default-check-out]
[-default_a=default-action] [-default_n=default-naming] [-default_t=default-item-type] [-
asso=associated-directory] [-copy_a=copy-associated-files] [-copy_n=copy-non-master-type]
[-default_o=default-owner] [-l=load-log-file] [-s=save-log-file] [-auto=translate_mode] [-
propagate=yes/no] [-export_dfa_kf=dfa_only|dva_in_part] [-export_dfa_list=] [-rev_up=] [-
attach_log_file=] [-copy_n=copy-non-master-type] [-dr=dryrun] [-h=help]
```

ARGUMENTS

-pim

Set **-pim** to **Yes** to initialize Teamcenter Integration for NX only, instead of native NX.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-corba_ior_file

Specifies the Teamcenter password server IOR file.

-http_url

Specifies the HTTP URL for four-tier configuration.

-h

Displays help for this utility.

RESTRICTIONS

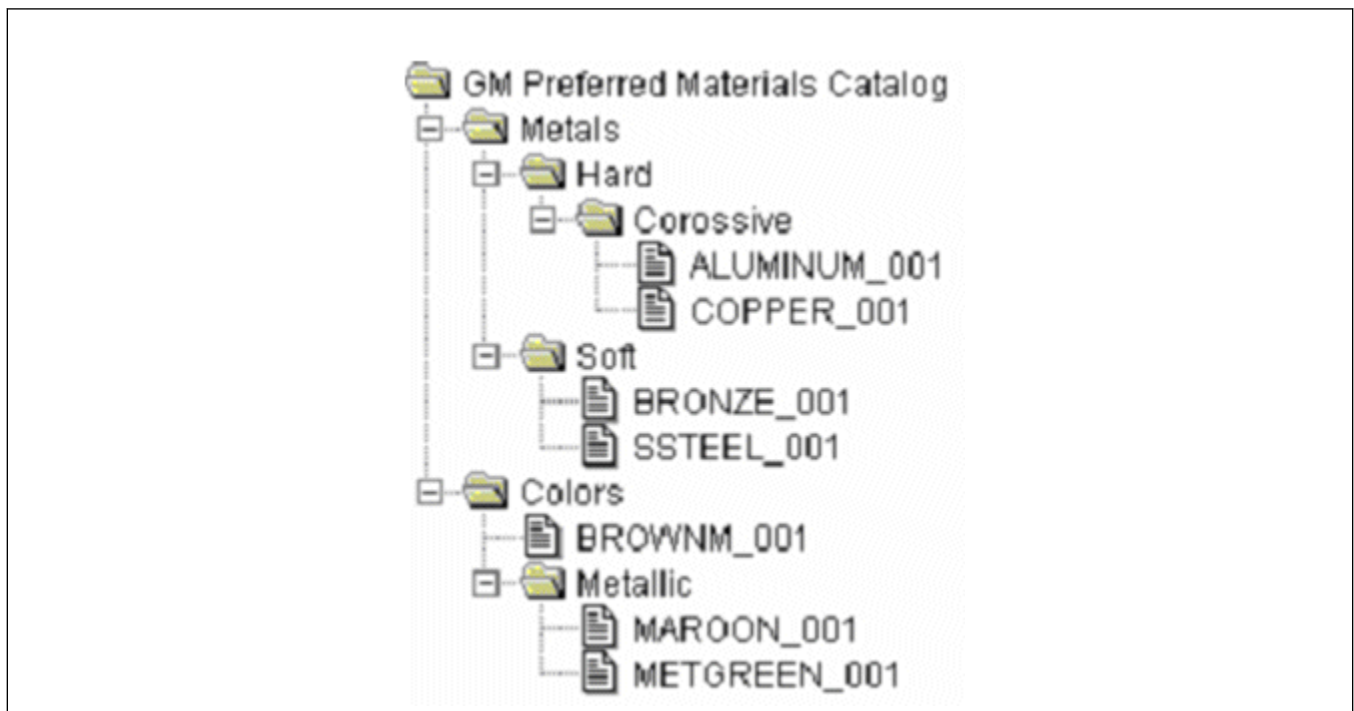
None.

gmo_create_material_form_templates

Creates new material form templates by reading the input from an ASCII text file. Input must be supplied in a defined format, as specified below. Material form templates are of the form type **Material** and are stored in folders or subfolders of the folder type **Material Template**. These folders and forms are stored in the **GM Preferred Materials Catalog** that is displayed in the Teamcenter administrative user's **Home** folder.

INPUT FILE FORMAT

This section describes the input file format using the sample folder structure shown in the following figure.



Sample directory structure

To achieve the structure shown in the sample directory structure, the input file must be in the following format:

```

<Folder1>[:<Folder2>:<Folder3>:<Folder4>:...]$<Form Name>#value of p_mat
# value of p_pcoat# value of p_perf# value of p_pcperf# value of

p_appear#
\value of p_fin# value of p_svc# value of p_addreq# value of p_mateng
# value of

p_appeng# value of p_pnteng
  
```

Folders are delimited by a colon (:), folders and forms are delimited by a dollar sign (\$), and the form and form values are delimited by the (#) symbol.

Note:

The delimiting symbols described in the previous paragraph assume that these symbols are not used as values in any of the materials form fields.

The following is an example of the input file format.

```
Metals:Hard:Corossive$ALUMINUM_001#p_mat#p_pcoat#p_perf#p_pcperf
#p_appear#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Metals:Hard:Corossive$COPPER_001#p_mat#p_pcoat#p_perf#p_pcperf#p_appear
#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Metals:Soft$BRONZE_001#p_mat#p_pcoat#p_perf#p_pcperf#p_appear
#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Metals:Soft$SSTEEL_001#p_mat#p_pcoat#p_perf#p_pcperf#p_appear
#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Colors$BROWNM_001#p_mat#p_pcoat#p_perf#p_pcperf#p_appear
#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Colors:Metallic$MAROON_001#p_mat#p_pcoat#p_perf#p_pcperf#p_appear
#p_fin#p_svc#p_addreq#p_mateng#p_appeng#p_pnteng
Colors:Metallic$METGREEN_001#metallic#green#high#corrosive#greenish
#smooth#requires applying grease#none#user#user#user
```

SYNTAX

```
gmo_create_material_form_templates [-u=user-id {-p=password | -pf=password-file}
-g=group]
[-infile=full-path-to-input-file] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see *Manage password files*.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-infile

Specifies the full path to the input file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*. In addition, the **gmo_create_material_form_templates.log** file is created in the directory from which the utility is run.

RESTRICTIONS

None.

EXAMPLES

- To obtain help for this utility, enter the following command on a single line:

```
gmo_create_material_form_templates -h
```

- To read the supplied input file and create folders and forms that are inserted in the **GM Preferred Materials Catalog** folder, enter the following command on a single line:

```
$GMPDM_ROOT/bin/gmo_create_material_form_templates -u=Tc-admin-user
```

```
-p=password -g=group -infile=/tmp/MATERIAL.txt
```

gmo_find_changed_install_assem

Locates the installation assemblies that have changed since the specified date and that are configured with the specified revision rules.

SYNTAX

```
gmo_find_changed_install_assem [-u=user-id {-p=password | -pf=password-file}
-g=group]-time=date-time -revision_rule=rule1
[-revision_rule=rule2 ...-revision_rule=rulen] -rev_rule_file=file-name
[-obj_type=object-name] [-out_file=output-file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-time

Specifies the date and time from which the **Released CORP_Install** item revisions are to be searched. This should be provided in the Teamcenter-specified format. An operating system file name can also be given whose last modification time will be taken as the time.

-revision_rule

Specifies revision rules for which the search is to be done. You can specify this argument multiple times. If this is given, the **rev_rule_file** argument cannot be used.

-rev_rule_file

Specifies a text file listing all the revision rules on separate lines. If this argument is given, **revision_rule** option cannot be used.

-obj_type

Specifies the type of object which is to be searched for the change in release status since the specified date. This argument is optional and defaults to objects of type **CORP_Install Revision**.

-out_file

Specifies file to which output is written. This argument is optional. If it is not specified, the output is sent to **stdout**.

-h

Displays help for this utility.

gmo_get_partspec

Retrieves the part specification of the specified dataset.

SYNTAX

```
gmo_get_partspec  
-dataset_tag=dataset [-h]
```

ARGUMENTS

-dataset_tag

Specifies the dataset tag in a string.

-h

Displays help for this utility.

gmo_get_pds_info

Retrieves the **pds** attributes for the given part numbers. The output files are generated at the location specified by %TC_TMP_DIR % with the names **sitename_timestamp_parts_notfnd.txt** and **sitename_timestamp_parts_fnd.txt**.

SYNTAX

```
gmo_get_pds_info [-u=user-id {-p=password | -pf=password-file} -g=group]
[-input_file=input-file-name | -itemKeyFile=file-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-input_file

Specifies the input file with part numbers and revision IDs separated by the ~ (tilde) character.

-itemKeyFile

Specifies the input file name containing the keys of the desired part numbers. The file format is:

```
-key = [keyAttr1=keyVal1][keyAttr2=keyVal2]...  
-key =[keyAttr1=keyVal1][keyAttr2=keyVal2]...  
-key =[keyAttr1=keyVal1][keyAttr2='keyVal2']...
```

-h

Displays help for this utility.

gmo_install_usage_queries

Installs usage queries.

SYNTAX

```
gmo_install_usage_queries [-u=user-id {-p=password | -pf=password-file} -g=group]  
[-recreate] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-recreate

Optional parameter. If specified, currently installed usage queries are deleted prior to installation of corresponding GMO-specific usage queries. As default, usage queries are normally installed. Siemens Digital Industries Software recommends you use this parameter to avoid installation error

messages when attempting to install over existing queries. This utility can be executed repeatedly using this option.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_ipvbom_import

Imports build intent data from a PLM XML file and creates a change object of type **GM Build Intent**.

SYNTAX

```
gmo_ipvbom_import [-u=user-id {-p=password | -pf=password-file} -g=group]  
-xml_file=full-path-to-PLM XML-file [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-xml_file

Specifies the path to the PLM XML input file containing the build intents.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility must be used only to import build intent data. It is not intended to import other data contained in PLM XML files.

EXAMPLES

- To obtain help for this utility, enter the following command on a single line:

```
gmo_ipvbom_import -h
```

- To import build intent data contained in the **/tmp/IMPORT.xml** file, enter the following command on a single line:

```
gmo_ipvbom_import -u=Tc-admin-user -p=password -g=group -xml_file=  
/tmp/IMPORT.xml
```

gmo_ipvbom_export

Exports specific build intent information, all build intent information, specific build intents along with partial build intents, full BOM or incremental BOM data.

SYNTAX

```
gmo_ipvbom_export [-u=user-id {-p=password | -pf=password-file} -g=group]
[-build_id=build-intent-id-number] [-fullbom=yes | no] [-inputfile=full-path-to-inputfile]
-xml_path=full-path-to-PLM XML-file [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-build_id

Specifies the ID number of the build intent to be exported.

-fullbom

Specifies whether or not to export the full BOM. Valid values are **yes** or **no**. If **yes**, the utility exports the full BOM; otherwise, the delta BOM is exported.

The default value is **yes**.

-inputfile

Full path to file.

-xml_path

Specifies the path to the output directory containing the XML file. If not specified, the path defined in the preference file is used.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*. In addition, the **IPVBOM_build_intent_status** preference, which specifies the release status used to obsolete older revisions of the build intent changes, must be set.

FILES

As specified in *Log files produced by Teamcenter*. In addition, the values of the **IPVBOM_compare_mode_var_level_RevisionCompare_ItemTypes** and **IPVBOM_compare_mode_var_level_Occurrence_Notes** preferences affect the behavior of the **gmiman_export** utility. For more information about these preferences, see the *Teamcenter Environment Variables*.

RESTRICTIONS

This utility must be used only to export build intent data. It is not intended to export other data to PLM XML files.

EXAMPLES

- To obtain help for this utility, enter the following command on a single line:

```
gmo_ipvbom_export -h
```

- To export build BOMs listed in the **/tmp/EXPORTS.txt** file from the Teamcenter database to a PLM XML file, enter the following command on a single line:

```
gmo_ipvbom_export -u=Tc-admin-user -p=password -g=group -fullbom=yes  
-xml_path=/tmp -inputfile=/tmp/EXPORT.txt
```


gmo_ipvbom_pulldate

Updates the pull date information for each build intent that is defined in a PLM XML file.

SYNTAX

```
gmo_ipvbom_pulldate [-u=user-id {-p=password | -pf=password-file} -g=group]  
[-xml_file=absolute-path-of-xml-file] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-xml_file

Specifies the full path of the PLM XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To obtain help for this utility, enter the following command on a single line:

```
gmo_ipvbom_pulldate -h
```

- To update pulldate information for each build intent contained in the **/tmp/PULLDATE.xml** file, enter the following command on a single line:

```
gmo_ipvbom_pulldate -u=Tc-admin-user -p=password -g=group -xml_file=c:/tmp/  
PULLDATE.xml
```

gmo_migrate_ulink_to_rdvauto

Migrates ULink occurrences in a VAS to GRDVA occurrences.

SYNTAX

```
gmo_migrate_ulink_to_rdvauto [-u=user-id {-p=password | -pf=password-file}  
-g=group]  
[-product=product-item-id] -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...  
[,keyAttrN=keyValN]  
-rev=product-revision-id -ia_list=IA-item-id-file  
-process_path=y | -process_all=y | -process_rules=y  
-logfile=logfile-name [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-product

Specifies VAS item ID.

-key

Specifies the key of the VAS item. Use the following format:

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-rev

Specifies VAS revision ID.

-logfile

Specifies the name of the logging file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_migrate_usage_nves

Migrates existing architecture breakdowns to the newer, simplified format of named variant expression (NVE) comprising usage, year, and production usage. You should only run this utility if you created NVEs in Teamcenter 2007.1 MP6 or earlier by running the **rdv_import_usage** utility. If you created NVEs in later releases, it is not necessary to run this utility. Otherwise, you should run this migration utility only once on each architecture breakdown.

The utility outputs the usage NVEs in the architecture breakdown and the assembly structure in the new format of NVEs. It also generates a list of the older format NVEs that are replaced and you can use this list to identify unused NVEs for deletion.

For example, it takes an older format NVE coded as D9_2009-09_AA5M_1PD69_PS and generate the following NVE code strings:

```
D9_AA5M_1PD69_000
Year_2009-09
ProductionUsage
```

SYNTAX

```
gmo_migrate_usage_nves [-u=user-id {-p=password | -pf=password-file} -g=group]
-revision_rule=rev-rule-string -mode=migration-mode
-top_level_AB_id=vehicle-architecture-id | -louholder_item_id=item-id
-archId_list_file=file-name
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-revision_rule

Specifies a valid revision rule string to configure the structure window.

-mode

Specifies the running mode of the utility, either **report** to generate a report of NVEs to migrate or **migrate** to initiate migration of affected NVEs.

-top_level_AB_id

Specifies the item ID of a top level architecture. All LOUs under LOU holders in the specified architecture are processed. If you specify this argument, do not specify a **-louholder_item_id** argument.

-louholder_item_id

Specifies the item ID of a LOU holder. If you specify this argument, do not specify a **-top_level_AB_id** argument.

-archId_list_file

Specifies a flat file containing architecture IDs. The utility uses these architecture IDs to retrieve the NVEs of affected LOUs.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

RESTRICTIONS

None.

FILES

As specified in [Log files produced by Teamcenter](#).

EXAMPLES

To migrate all the NVEs for the **Model_2009_AB** architecture breakdown, enter the following command on a single line:

```
gmo_migrate_usage_nves -u=user-name -p=password -g=group  
-revision_rule=production -mode=migrate -top_level_AB_id=Model_2009_AB  
-archId_list_file=AB.txt
```

gmo_set_rel_status

Sets the release status for the item revisions and related datasets, revision masters, and BOMview revisions with the specified statuses. It searches for the specified folder in home folder of the Teamcenter administrative user and finds all the items and item revisions in that folder. The utility then sets the specified *product-release-status* for all the non-**PDE** item revisions, datasets, BOMview revisions, revision master forms and the specified *pdi-release-status* for all the **PDI** item revisions, datasets, BOMview revisions, and revision master forms. It sets the given release status only if no release status is set before for item revision, dataset, bomview revision, and revision master form.

SYNTAX

```
gmo_set_rel_status [-u=user-id {-p=password | -pf=password-file} -g=group]
folder_name prod_rel_status pdi_rel_status
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

RESTRICTIONS

The *folder_name* must be present in the home folder of the Teamcenter administrative user.

gmo_split_usage

Divides GM Corporate Dictionary (Architecture Breakdown Structure) and Line Of Usage Data (GPDS XML files) into predefined blocks of data (GPDS XML files). For each block of data, this utility creates script/batch file that is run to upload the Corporate Dictionary/Usage Data.

SYNTAX

```
gmo_split_usage [-u=user-id {-p=password | -pf=password-file} -g=group]
=input-file [{-log=name-of-logfile -max=maximum-usage |
[-enable_lock_grabbing] -arch=y
{-archtop_item_id=arch-top-item-id | -archtop_item_name=arch-item-name}}]
-import_usage_log=logfile-name -generate_delta_xml [
-enable_lock_grabbing] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-input

Specifies the name of the input file.

-log

Specifies the name of the log file.

-max

Specifies the maximum usage.

-arch

Boolean flag to upload Corporate Dictionary.

-archtop_item_id

Specifies item ID of the Architecture Breakdown.

-archtop_item_name

Specifies the name of the architecture top item.

-import_usage_log

Specifies for each block of data, a separate logfile is generated with increments indicating each data block.

-generate_delta_xml

Specifies the utility is to generate the Delta XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_update_vas_data

Connects to the GPDS system and updates the VAS registration.

SYNTAX

```
gmo_update_vas_data [-u=user-id {-p=password | -pf=password-file} -g=group]
-datasource=external-datasource-name [-hostname=external-datasource-hostname]
[-user=external-proxy-user] [-passwd=external-proxy-password] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-datasource

Specifies the name of the external datasource.

-hostname

Specifies the hostname of the external datasource.

-user

Specifies the user of the external datasource.

-passwd

Specifies the password of the external datasource.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_upgrade_dlist_objects

Migrates the distribution list objects created as a dataset. It also searches for all distribution list objects existing in the database and transforms them into **EPMAssignmentList** objects.

DESCRIPTION

SYNTAX

```
gmo_upgrade_dlist_objects [-u=user-id {-p=password | -pf=password-file} -g=group]
[-delete_old] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-delete_old

Indicates that the utility is to delete the old assignment list objects after they are upgraded.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_validate_xml

Validates the GPDS generated XML file. It ensures that all of the model option statements and model designators contain the correct VDS records and the corporate dictionary is correctly defined.

SYNTAX

```
gmo_validate_xml [-u=user-id {-p=password | -pf=password-file} -g=group]  
-input=input-xml-file-name [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-input

Specifies the name of the XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

gmo_vds_util

Synchronizes the variant data stored with the item revision between the base and multiple target item revisions.

SYNTAX

```
gmo_vds_util [-u=user-id {-p=password | -pf=password-file} -g=group]
[-i=item-id
| -key=[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]]
-b=base-item-rev [-t=target-item-rev1 ... ] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-i

Specifies the item ID.

-key

Specifies the key of the item. Use the following format:

`[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN]`

To find the key of an object, use the **get_key_string** utility.

For more information, see *BMIDE for Data Model Design*.

-b

Specifies the base item revision.

-t

Specifies the first target item revision. You can specify this argument multiple times.

-h

Displays help for this utility.

rdv_import_usage

Imports usages from an XML file to Teamcenter usage representations. The utility applies all model NVEs available during the pre-usage stage to the top-level architecture node. The model NVEs are not applied to the children of the top-level architecture node (the architecture breakdown elements). However, the stored model NVEs on the top-level architecture element are available for manual application to any children in the architecture breakdown. When working in the Replace Design in Product wizard in Design Context or Structure Manager, the user can select the model NVEs from the top-level architecture breakdown, rather than from the preselected architecture breakdown element (ABE).

Unlike in earlier versions of Teamcenter, model NVEs are not associated with the ABEs under the top-level architecture breakdown. Rather, model NVEs are stored as absolute occurrence data on the top-level architecture node. The utility creates the necessary absolute occurrence data at the top-level during the pre-usage import step.

Platform Designer allows the user to view model and manual NVEs associated with the top-level. You can use the **Add** command to add associated manual NVEs from the variant expression block at the top level to the absolute occurrence data.

Optionally, you can configure the utility to retry if it fails to complete a usage load for any reason on the first attempt. Failure to complete the load impacts downstream processes, as users cannot align their CAD solutions to the most recent PLM system changes. The utility retries obtaining a lock on the objects needing modification, typically, the product revision or top-level architecture node. Once a lock is obtained, the utility completes usage load operations. The optional ability for the utility to obtain the necessary locks is set with the **RDV_enable_product_lock** preference.

Caution:

Data loss may occur if the utility removes the lock while a user is actively editing data associated with the top-level product.

In the event of a failure while importing LOUs, it reports error codes and descriptions in the system log. These reports can be interpreted by users or scripts to take the necessary corrective action.

Note:

The format of the NVEs created by this utility changed with effect from Teamcenter 2007.1 MP6. Newer versions of the utility automatically include options and values so that you can use those options for the manual creation of NVEs and saved variant rules (SVRs) in Platform Designer. While running the utility, Teamcenter checks to see if the item ID of the product item is present in the **PSM_global_option_item_ids** preference. If so, it automatically creates the variability on the top level architecture breakdown. The import utility then applies variability for every option-value on the architecture breakdown (VAB). During execution of the **rdv_import_usage** utility, Teamcenter does not enforce hierarchical variability on the architecture breakdown (VAB).

If you used an earlier version of this utility to create NVEs, you should run the **gmo_migrate_usage_nves** utility to migrate them to the new format. If you created NVEs in

Teamcenter 2007.1 MP6 or later, it is not necessary to migrate them. Examples of the old and new formats follow:

Old NVE format	New NVE format
D1_2008-UP_8619_2WC69_P	D1_8619_2WC69_000Year_2008-UPProductionUsage
D1_2007-07_8619_2WR69_P	D1_8619_2WR69_000Year_2007-07ProductionUsage
D1_2007-07_8619_2WP69_P	D1_8619_2WP69_000Year_2007-07ProductionUsage

A sequence number is appended to the authorized NVE so that true availability changes in model codes across years may be recorded in the NVE content. In the previous examples, the sequence number is **000**.

SYNTAX

```
rdv_import_usage [-u=user-id {-p=password | -pf=password-file} -g=group]
-input_file=name-of-xml-file -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-input_file

Specifies the XML input file from which the utility imports usages.

-h

Displays help for this utility.

ENVIRONMENT

- As specified in *Setting up a Teamcenter command-line environment*.
- The **TC_retry_time** preference determines the time interval at which the utility tries to obtain a lock on objects to import, if it is not initially successful. You can set this preference as an environment variable so that the default value can be overridden by scripts during usage loading.
- The **TC_max_number_of_retries** preference determines how many attempts the utility makes to obtain a lock on objects to import, if it is not initially successful. You can set this preference as an environment variable so that the default value can be overridden by scripts during usage loading.

RESTRICTIONS

None.

FILES

As specified in *Log files produced by Teamcenter*.

EXAMPLES

To import usages, enter the following command on a single line:

```
rdv_import_usage -u=user-name -p=password -g=group -input_file=pdis101.x
```

The utility uses the **RDV_IMPORT_USAGE_TM** transfer mode to convert the XML file to PLM XML format, via a style sheet.



17. Aerospace and Defense utilities

default_adsfoundation_queries

Installs the saved queries in the ADS Foundation template with names and descriptions either in the locale specified for the system or in all supported locales. The following queries are installed by this utility:

- Find ADSTechDocument
- Find ADSDrawing
- Find ADSPart
- Find ADSDesign

Note:

This utility runs automatically when the Aerospace and Defense solution is installed or upgraded.

SYNTAX

```
default_adsfoundation_queries [-u=user-id {-p=password | -pf=password-file}
-g=group]-locales=locale-code | ALL [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-locales

Specifies the locale, using locale codes or **ALL**, for which translated query names and descriptions are installed. You can specify a single locale, or you can specify multiple locales in a comma-separated list, for example, **en_US,de_DE,fr_FR**. Using the **ALL** value installs all locales supported by your Teamcenter system.

For a list of locale codes, see *Teamcenter Localization*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in [Setting up a Teamcenter command-line environment](#).

FILES

As specified in [Log files produced by Teamcenter](#).

RESTRICTIONS

None.

EXAMPLES

- To install the ADS Foundation saved queries with names and descriptions in all locales supported for the system, enter the following command on a single line:

```
default_adsfoundation_queries -u=Tc-admin-user -p=password -g=group
- locales=ALL
```

- To install the ADS Foundation saved queries with names and descriptions in English, German, and Czech, enter the following command on a single line:

```
default_adsfoundation_queries -u=Tc-admin-user -p=password -g=group
- locales=en_US,de_DE,cs_CZ
```

default_adschangemanagement_queries

Installs the ADS Change Management saved queries with names and descriptions in either the locale specified for the system or in all supported locales. The following queries are installed by this utility:

- **Find All Change Notice Revisions**

Note:

This utility runs automatically when the Aerospace and Defense Change Management solution is installed or upgraded.

SYNTAX

```
default_adschangemanagement_queries [-u=user-id {-p=password |  
-pf=password-file} -g=group]  
-locales=locale-code | ALL [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-locales

Specifies the locale, using locale codes or **ALL**, for which translated query names and descriptions are installed. You can specify a single locale or you can specify multiple locales in a comma-separated list, for example **en_US,de_DE,fr_FR**. Using the **ALL** value installs all locales supported by your Teamcenter system.

For a list of locale codes, see *Teamcenter Localization*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To install the ADS Change Management saved queries with names and descriptions in all locales supported for the system, enter the following command on a single line:

```
default_adschangemanagement_queries -u=Tc-admin-user -p=password -g=group
-locales=ALL
```

- To install the ADS Change Management saved queries with names and descriptions in English, German, and Czech, enter the following command on a single line:

```
default_adschangemanagement_queries -u=Tc-admin-user -p=password -g=group
-locales=en_US,de_DE,cs_CZ
```

update_locationcode_from_owningorg

Updates the **Original Location Code** of items and **Current Location Code** of item revisions based on the **Organization ID** of the organization associated with the item or item revision.

This utility also creates a relation between groups and company locations based on the defined organization structure.

SYNTAX

```
update_locationcode_from_owningorg [-u=user-id {-p=password | -pf=password-file}
-g=group] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To update the **Original Location Code** of all items and **Current Location Code** of all item revisions based on the **Organization ID** attribute, and create relations between groups and company locations:

```
update_locationcode_from_owningorg -u=Tc-admin-user -p=password -g=group
```




18. Running CAE utilities

cae_configure_dm_propertyset

This utility reads the **NodeXMLConfig.xml** file and configures the property set for data mapping.

The **NodeXMLConfig.xml** configuration XML file is used by the simulation administrator to control which items, item revisions, forms, or BOM line attributes are processed. The administrator can edit the required attributes in the XML file in such a way that Teamcenter processes only those needed attributes. This significantly improves the performance and memory usage of data map and structure map execution operations.

For more information about configuring data map and structure map rules using a configuration file, see *Simulation Process and Data Management — Deployment and Administration*.

Property sets provide a nonprogrammatic way to control what is placed in the **UserData** element. The **UserData** element is a container for a list of name-value pairs that allows any attribute or property that is not in the PLM XML equivalent to be stored. Property sets are a way to get Teamcenter data into the PLM XML file that may not be in the PLM XML schema and can be controlled by a programmer who does not know Teamcenter code or how to write it. A property set is simply a list of property set clauses.

For more information about transfer modes, closure rules, and property sets, see *Tc XML and PLM XML Configuration for Data Import and Export*.

If the **Materials Management** template is not installed and when you run this utility, the following error is displayed on the Teamcenter command prompt after execution:

The PropertySet is configured with partial errors. There are invalid object classes, types or properties found in the NodeXMLConfig.xml. Please execute the utility with "-log" option and check log file for more details.

This error message can be ignored. To avoid seeing this message, comment out the **smn:NODE_LINE** attributes in the **NodeXMLConfig.xml** file as follows:

Example:

```
<!-- <smn:NODE_LINE CLASS="Mat1MaterialRevision"
TYPE="Mat1MaterialRevision">
    <smn:ATTR_NODES>
        <smn:ATTR_NODE INTERNAL_NAME="object_name" NAME="Name"/>
        <smn:ATTR_NODE INTERNAL_NAME="object_desc"
NAME="Description"/>
        <smn:ATTR_NODE INTERNAL_NAME="item_id" NAME="ID"/>
        <smn:ATTR_NODE INTERNAL_NAME="structure_revisions"
NAME="BOM View Revisions"/>
        <smn:ATTR_NODE INTERNAL_NAME="project_ids"
NAME="Project IDs"/>
    </smn:ATTR_NODES>
```



```

    <smn:NODE_LINKS>
      <smn:NODE_LINK CLASS="CAEGeometryRevision"
        RELATIONSHIP_TYPE="Mat1UsesMaterial"
        TYPE="CAEGeometryRevision"/>
    </smn:NODE_LINKS>
  </smn:NODE_LINE> -->

```

SYNTAX

cae_configure_dm_propertyset

```

[-u=user_name]
[-p=user_name]
[-pf=password-file]
[-g=dba]
[-log=log-file-name-containing-results-of-utility-execution]
[-h]

```

ARGUMENTS

-u

Specifies the user ID.

This is generally a DBA user or a user from the Simulation Admin group.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-log

(Optional) Specify the name of the log file containing results of the utility execution.

-h

Displays help for this utility.

cae_execute_cae_accountability_check

Compares the attributes of a product and model structure using the accountability check framework along with data mapping rules. The attributes are compared by applying data mapping rules. The attributes marked as **mapped** in the **NodeXMLConfig.xml** file used by data mapping is considered for comparison. Data mapping rules are applied on the input product structure, and the data mapped values are compared with the model attribute value. The product and model item keys and the revision IDs are required along with the data mapping domain as input. You can optionally specify the configuration information for the product and the model. The result of the comparison is created as a named reference containing a dataset (Microsoft Excel file) with the name you specify in the **-comparison_dataset_name** argument. The result is attached to the root of the model structure.

SYNTAX

```
cae_execute_cae_accountability_check
[-u=user_name
{-p=user_name
-pf=password-file}
-g=dba]
-ou=cae_admin
-og=simulation_administrator
-or=simulation_administrator
-product_item_key=key-to-the-root-item-of-the-product-structure
-product_rev_id=revision-ID-of-the-product-item-revision
-product_rev_rule=revision-rule-of-the-product-structure
-product_snapshot=snapshot-folder-name-for-product-structure
-product_variant_rule=variant-rule-for-the-product-structure
-model_item_key=key-to-the-root-item-of-the-model-structure
-model_rev_id=revision-ID-of-the-model-item-revision
-model_rev_rule=revision-rule-of-the-model-structure
-model_snapshot=snapshot-folder-name-for-the-model-structure
-model_variant_rule=variant-rule-of-the-model-structure
-domain=data-mapping-domain-you-want-to-use
-mode=compare-or-propagate-mode
-comparison_dataset_name=dataset-name-to-save-the-comparison-log-if-
the-mode-is-compare
-param_file=parameter-file-in-text-format-containing-parameters-for-the-utility
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ou

(Optional) Specifies the owning user ID.

This user owns all created objects.

-og

(Optional) Specifies the group associated with the owning user.

Note:

If you specify the **-ou** argument and do not specify the **-og** argument, the user's default group is used.

-or

(Optional) Specifies the role associated with the owning user.

-product_item_key

Specifies the key to the root item of the product structure.

Example:

```
-product_item_key=\"item_id=000001,my_prop=N VH\\\"\\n\" );
```

-product_rev_id

Specifies the revision ID of the product item revision.

-product_rev_rule

(Optional) Specifies the revision rule of the product structure.

This argument is not considered if you specify the **-product_snapshot** argument.

-product_snapshot

(Optional) Specifies the snapshot folder name for product structure.

-product_variant_rule

(Optional) Specifies the variant rule for the product structure.

-model_item_key

Specifies the key to the root item of the model structure.

Example:

```
-model_item_key=\"item_id=000002,my_prop=NVHM\\\"\\n\" );
```

-model_rev_id

Specifies the revision ID of the model item revision.

-model_rev_rule

(Optional) Specifies the revision rule of the model structure.

This argument is not considered if you specify the **-model_snapshot** argument.

-model_snapshot

(Optional) Specifies the snapshot folder name for the model structure.

-model_variant_rule

(Optional) Specifies the variant rule of the model structure.

-domain

Specifies the data mapping domain you want to use.

-mode

Specifies the mode for CAE accountability check. The mode can be **compare** or **propagate**.

If you use the **compare** mode, the resultant Excel dataset is created under the model root item revision with the name given in the **comparison_dataset_name** dataset name.

If you use the **propagate** mode, the resultant propagate HTML dataset is created under the model root item revision with the name from the **CAE_attribute_propagate_summary_dataset_name** preference.

-comparison_dataset_name

Specifies the dataset name to save the comparison log to if the mode is **compare**.

If the dataset name already exists in the model revision, the dataset content is replaced.

-param_file

Specifies a parameter file in text format containing parameters for the utility.

-h

Displays help for this utility.

EXAMPLE 1

Using an item key and rev ID (mode is **compare**), enter the following command in a single line:

```
cae_execute_cae_accountability_check
-u=adminjones
-p=adminjones
-g=admin
-product_item_key=item_id=0001
-product_rev_id=A
-product_rev_rule=myRevRule
-product_variant_rule=myVariantRule
-model_item_key=item_id=0002
-model_rev_id=A
-model_rev_rule=myRevRule
-model_variant_rule=myVariantRule
-domain=CAE
-mode=compare
-compare_log=C:\temp\comparereport.html
```

EXAMPLE 2

Using a product and model snapshot (mode is **propagate**), enter the following command in a single line:

```
cae_execute_cae_accountability_check
-u=adminjones
-p=adminjones
-g=admin
-product_snapshot=prodSnapshot
-product_variant_rule=myVariantRule
-model_snapshot=ModelSnapshot
-model_variant_rule=myVariantRule
-domain=CAE
-mode=propagate
```

EXAMPLE 3

Using a parameter file (mode is **compare**), enter the following command in a single line:

```
cae_execute_cae_accountability_check  
-u=adminjones  
-p=adminjones  
-g=admin  
-param_file=C:\temp\myparamfile.txt
```

Parameter file contents:

```
product_item_key=item_id=0001  
-product_rev_id=A  
-product_rev_rule=myRevRule  
-product_variant_rule=myVariantRule  
-model_item_key=item_id=0002  
-model_rev_id=A  
-model_rev_rule=myRevRule  
-model_variant_rule=myVariantRule  
-domain=CAE  
-mode=compare  
-comparsion_dataset_name=ComparisonReport
```

cae_execute_datamap

Applies data mapping rules to an input structure by providing the root item revision of the input structure and the configuration information of the structure along with the domain for the data mapping rules.

When a snapshot is produced, the **Snapshot** folder, rather than the root item of the resulting structure, is pasted on the invoking user's **Newstuff** folder.

SYNTAX

```
cae_execute_datamap
[-u=cae_analyst
{-p=password
-pf=password-file}
-g=CAE_designer]
-inputItemKey=key-to-the-root-item-of-the-input-structure
-revID=revision-ID-of-the-input-item-revision
-snapshotOutput=snapshot-folder
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputItemKey

Specifies the key to the root item of the input structure; for example, **000001**.

-revID

Specifies the revision ID of the input item revision.

-snapshot

Specifies the snapshot folder name.

-h

Displays help for this utility.

EXAMPLES

To execute a data map on a input structure, enter the following command on a single line:

Example 1:

```
cae_execute_datamap
-u=cae_analyst
-p=password
-g=CAE_designer
-inputItemKey=000001 (key-to-the-root-item-of-the-input-structure)
-revID=A (revision-ID-of-the-input-item-revision)
-snapshotOutput=snapshot-folder
```

Example 2:

```
cae_execute_datamap
-u=cae_analyst
-g=CAE_designer
-inputItemKey=000001 (key-to-the-root-item-of-the-input-structure)
-revID=A (revision-ID-of-the-input-item-revision)
```

cae_execute_derive_structures

Applies the derivative rule to an input structure when the analyst provides the root item revision and the configuration information of the input structure. The output structures are pasted in the Teamcenter **NewStuff** folder.

SYNTAX

cae_execute_derive_structures

[-u=*user_name*]
 [-p=*password*]
 [-pf=*password-file*]
 [-g=*dba*]
 [-input_item_key=*key-of-the-root-item*]
 [-rev_id=*revision-ID-of-the-input-item-revision*]
 [-structure_config=*structure-configuration-name*]
 [-rule_name=*name-of-derivative-rule*]
 [-root_name=*optional-name-to-be-assigned-to-output-root*]
 [-root_index=*optional-index-to-be-assigned-to-output-root*]
 [-structure_count=*number-of-structures-to-be-created*]
 [-dr_item_key=*key-of-the-derivative-rule-item*]
 [-h]

ARGUMENTS

-u
 Specifies the user ID.

-p
 Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf
 Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-input_item_key

Specifies the key string to the root item of the input structure.

For example: **-input_item_key=000021** where *000021* is the item ID.

-rev_id

Specifies the revision ID of the input item revision.

-structure_config

Specifies the structure configuration name.

If this argument is not used, the system assumes the default configuration settings.

-rule_name

Specifies the name of the derivative rule you want to apply.

-root_name

Specifies the optional name you want to assign to the output root.

-root_index

Specifies the optional index you want to assign to the output root.

-structure_count

Specifies the number of structures you want the system to create. If this argument is not used, the system assumes the number as 1.

-dr_item_key=

Specifies the key of the derivative rule item.

-h

Displays help for this utility.

cae_execute_structuremap

Applies structure map rules to an input structure by providing the root item revision of the input structure and the structure map item revision containing the rule along with the configuration information of the structure.

When a snapshot is produced, the **Snapshot** folder, rather than the root item of the resulting structure, is pasted on the invoking user's **Newstuff** folder.

SYNTAX

```
cae_execute_structuremap
[-u=user-id
{-p=password
-pf=password-file}
-g=group]
-inputItemKey= key-of-the-root-item
-revID= revision-ID-of-the-input-item-revision
-SMItemKey= structure-map-item-key
-SMRev= structure-map-item-revision-ID
-snapshotOutput= snapshot-folder
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputItemKey

Specifies the key of the root item of the input structure; for example, **000001**.

-revID

Specifies the revision ID of the input item revision.

-SMItemKey

Specifies the structure map item key; for example, **000002**.

-SMRev

Specifies the structure map item revision ID.

-snapshotOutput

Specifies the output snapshot name of the output model item revision.

-h

Displays help for this utility.

EXAMPLES

To execute structure map rules on a input structure, enter the following command on a single line:

Example 1:

```
cae_execute_structuremap
-u=cae_analyst
-g=CAE_designer
-inputItemKey=000001 (key-to-the-root-item-of-the-input-structure)
-revID=A (revision-ID-of-the-input-item-revision)
-SMItemKey=000002 (structure-map-item-key)
-SMRev=B (structure-map-item-revision-ID)
-snapshotOutput=snapshot-folder
```

Example 2:

```
cae_execute_structuremap
-u=cae_analyst
-g=CAE_designer
-inputItemKey=000001 (key-to-the-root-item-of-the-input-structure)
-revID=A (revision-ID-of-the-input-item-revision)
-SMItemKey=000002 (structure-map-item-key)
-SMRev=B (structure-map-item-revision-ID)
```

cae_manage_datamap_definition

Updates the configured data map definition dataset using the specified **datamapping.xml** and the **NodeXMLConfig.xml** configuration files.

SYNTAX

cae_manage_datamap_definition

-u=*user-id*
-p=*password*
-pf=*password-file*
-g=*group*
-ou=*owning-user-id*
-og=*owning-group-name*
-migrate=*migrate-data-map-definition-files*
-datamap_file_path=*operating-system-path-to-data-mapping-file*
-nodexml_file_path=*operating-system-path-to-NodeXMLConfig-file*
-revise=*create-new-revision-of-configured-item*
-configure_propertyset=*configure-property-set-by-reading-NodeXMLConfig-file*
-configure_propertyset_log=*specify-path-to-log-file-generated-by-property-set-configuration*

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ou

(Optional) Specifies the ID of the owning user of the newly created objects.

-og

(Optional) Specifies the group associated with the owning user.

Note:

If used without a value, the default owning user's group is used.

-migrate

(Optional) Specifies an option to migrate the data map definition files located in the *TC_DATA* directory.

-datamap_file_path

(Optional) Specifies the operating system path to the data mapping file.

The file must exist at the specified location and the user must have access to the specified file. If the file path has spaces, use double quotation marks to specify the path.

-nodexml_file_path

(Optional) Specifies the operating system path to the **NodeXMLConfig.xml** configuration file.

The file must exist at the specified location, and the user must have access to the specified file. If the file path has spaces, use double quotation marks to specify the path.

-revise

(Optional) Creates a new revision of the configured item specified by the **CAE_datamap_files_location** preference value and uses the newly created revision to create the data mapping dataset and imports the files.

-configure_propertyset

(Optional) Configures the property set by reading the **NodeXMLConfig.xml** configuration file at the configured location.

-configure_propertyset_log

(Optional) Specifies a path to the log file generated as a part of the property set configuration.

Note:

This option is ignored if used without the **-configure_propertyset_log** option.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

cae_migrate_atl_preferences

Migrates legacy tool configuration settings to a new configuration managed by the Teamcenter vaulted dataset.

SYNTAX

cae_migrate_atl_preferences

[-u=user-id
{-p=password
-pf=password-file}
-g=group]
[-file=file-name (including file path)]
[-overwrite]
[-h]

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the path and file name of the tool preferences file containing tool preferences data to be migrated/appended to dataset-managed preferences.

If you do not specify this option, the utility looks for authoring tool launch preferences (used in Teamcenter 2007.1.x) in the database and simulation tool configuration (used in Teamcenter 8.0.x) in the **espf_configuration.xml** file in the **TC_DATA** folder and migrates them (if they are available) to a new configuration managed by the Teamcenter vaulted dataset.

Note:

You must specify the file name and the complete file path for the **-file** option.

-overwrite

Overwrites the existing tool preferences with legacy tool preferences.

If you do not specify this option, the utility appends legacy tool preferences to the existing tool preferences.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Migrate x preferences to the current configuration file in a dataset.

Note:

This example assumes there is no **espf_configuration.xml** file in the active **TC_DATA** folder and no **CAESolution** dataset with the name indicated by **CAE_simulation_tool_config_dsname** exists in the database.

```
cae_migrate_atl_preferences
-u=user_name
```

```
-p=*****
-g=dba
```

The system reads the configuration definitions from the **CAE_pre_processor_***, **CAE_solver_***, and **CAE_post_processor_*** preferences and creates an appropriate **espf_configuration.xml** file. The system creates a new **CAESolution** dataset named according to the value in **CAE_simulation_tool_config_dsname**, and imports the file as an **XML-File** reference.

- Migrate the x **espf_configuration.xml** file to the current configuration file in a dataset.

Note:

This example assumes a **simadmin** user is assigned to a **Simulation-Administrator** role in the **Simulation-Administration** group. This example assumes that there is an **espf_configuration.xml** file in the active **TC_DATA** folder, but no **CAESolution** dataset with the name indicated by **CAE_simulation_tool_config_dsname** exists in the database.

```
cae_migrate_atl_preferences
-u=simadmin
-p=*****
-g=Simulation-Administration
```

The system locates the **espf_configuration.xml** file in the active **TC_DATA** folder. The system creates a new **CAESolution** dataset named according to the value in **CAE_simulation_tool_config_dsname** and imports the file as an **XML-File** reference.

- Migrate the previous x **espf_configuration.xml** file to the current configuration file in a dataset.

Note:

This example assumes that there exists a **simadmin** user assigned to a **Simulation-Administrator** role in the **Simulation-Administration** group.

```
cae_migrate_atl_preferences
-u=simadmin
-p=*****
-g=Simulation-Administration
-file=D:\MyFiles\espf_configuration.xml
-overwrite
```

The system locates the **espf_configuration.xml** file using the provided path. The system attempts to locate a **CAESolution** dataset named according to the value in **CAE_simulation_tool_config_dsname**. If an appropriate dataset is found, the system imports the file as an XML file reference, overwriting any existing reference. If an appropriate dataset is not found, the system creates a new **CAESolution** dataset named according to the value in **CAE_simulation_tool_config_dsname** and imports the provided file as an **XML-File** reference.

cae_migrate_tool_configuration

Prior to Teamcenter 10.1.3, configured simulation tools were managed in the database in an XML file. From 10.1.3, simulation tools are managed in the database as individual business objects, and, therefore, you must migrate simulation tools when you upgrade to Teamcenter 10.1.3 and beyond.

After Teamcenter is migrated to a new version, you must run the **cae_migrate_tool_configuration** utility to migrate existing simulation tools.

This utility:

- Migrates the simulation tool configuration from the **espf_configuration.xml** file stored in the dataset identified by the **CAE_simulation_tool_config_dsname** preference.
- Imports the simulation tool configuration structure from the PLM XML **.xml** file at the target site.
- Exports the simulation tool configuration structure to the PLM XML **.xml** file at the source site.

Note:

Non DBA users must use the **-setRootToolByPass** option for migration or import operations.

If you want to export the unreleased tool BOM by using the **cae_migrate_tool_configuration** utility, set the value of the **PSEShowUnconfigdEffPref** preference to **true**.

SYNTAX

cae_migrate_tool_configuration

```
[-u=dba_user]
[-p=password]
[-pf=password-file]
[-g=dba]
[-ou=owning-user-id]
[-og=owning-group-name]
[-mode=export-or-import]
[-folder=folder-name]
[-fileName=PLM-XML-file]
[-lwofilename=ESPF-XML-file]
[-item_id=root-simulation-item-id]
[-rev=root-simulation-tool-item-rev-id]
[-rev_rule=revision-rule]
[-autorelease=auto-release-indicator]
[-setRootToolByPass=bypass-root-simulation-tool-setting]
[-h=displays-command-line-help]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ou

Specifies the owning user ID.

This user owns all created objects. This option is used only to migrate tool configuration from the XML file stored in the dataset.

-og

Specifies the group associated with the owning user.

Note:

If you specify the **-ou** argument and do not specify the **-og** argument, the user's default group is used.

-mode=EXPORT | IMPORT

Specifies the mode of this utility. If this option is not used, this utility migrates the tool configuration from the XML file stored in the dataset identified by the **CAE_simulation_tool_config_dsname** preference.

-mode=EXPO Exports the simulation tool structure in PLM XML format (.xml file) from the source
RT site.

-mode=IMPO Imports the simulation tool structure in PLM XML format (.xml file) from the target
RT site.

-folder=folder-name

Specifies the folder where the PLM XML and ESPF XML files are created. Required only for PLM XML mode of operation.

-fileName=PLM XML file

-mode=EXPO Specifies the PLM XML (.xml) file that contains information about the Simulation
RT Tool structure you want to export.

-mode=IMPO Specifies the PLM XML (.xml) file that contains information about the Simulation
RT Tool structure you want to import.

-lwofilename=ESPF-XML-file

-mode=EXPO Specifies the PLM XML (.xml) file that contains information about the LWOs in the
RT simulation tool structure you want to export.

-mode=IMPO Specifies the PLM XML (.xml) file contains information about the LWOs in the
RT simulation tool structure that needs to be imported.

-item_id=root-simulation-tool-item-id

Specify the item id of root simulation tool.

-rev=root-simulation-tool-item-rev-id

Specifies the item revision id of root simulation tool. This is required only for the PLM XML mode of operation.

-rev_rule=revision-rule

Specifies the revision rule to configure the tool structure. If not specified, the default revision rule is used. This is applicable for the PLM XML mode of operation.

-autorelease=auto-release-indicator

Releases the simulation tool structure after the import operation using the PLM XML mode of operation.

-setRootToolByPass

Bypasses the setting of the root simulation tool UID to the preference value to import or migrate from the XML file.

Note:

If this option is not used, the utility sets the root simulation tool to the preference value.

-h

Displays help for this utility.

EXAMPLES

1. Migrate the simulation tool configuration (**espf_configuration.xml**) file to the simulation tool structure and configure the simulation root tool.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=dba -ou=owning-user-id
```

2. Migrate the legacy tool configuration (**espf_configuration.xml** file) to the simulation tool structure, but do not configure the simulation root tool.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=dba -ou=owning-user-id  
-setRootToolByPass
```

3. Export the simulation tool configuration structure to the PLM XML (**.xml**) file using the default revision rule.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=dba -mode=EXPORT -  
folder=D:\Import-export\example_plmxml_folder -item_id=000082-rev=A
```

4. Export the simulation tool configuration structure to the PLM XML (**.xml**) file, using the **Any Status; Working** revision rule.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=dba -mode=EXPORT -  
folder=D:\Import-export\example_plmxml_folder -item_id=000082-rev=A -rev_rule=Any  
Status; Working
```

5. Import the simulation tool configuration structure from the PLM XML (**.xml**) file, using the default revision rule with the simulation administrator as the owning user, but do not configure the simulation root tool.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=simulation administrator  
-mode=IMPORT -folder=D:\Import-export\example_plmxml_folder -item_id=000082-rev=A -  
ou=owning-user-id -setRootToolByPass
```

6. Import the simulation tool configuration structure from the PLM XML (**.xml**) file, using the default revision rule, and release the structure after import, but do not configure the simulation root tool.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=DBA -mode=IMPORT  
-folder=D:\Import-export\example_plmxml_folder -item_id=000082-rev=A -autorelease  
-setRootToolByPass
```

7. Import the simulation tool configuration structure from the PLM XML (.xml) target site, release the structure, and configure the simulation root tool.

```
cae_migrate_tool_configuration -u=dba_user -p=password -g=DBA -mode=IMPORT -  
folder=D:\Import-export\example_plmxml_folder -item_id=000082-rev=A -autorelease
```


cae_validate_structuremap

Validates the **CAE Structure Map** item revision to ensure that the configured structure map rules are properly validated before they are executed.

SYNTAX

```
cae_validate_structuremap
[-u=user_name
{-p=user_name
-pf=password-file}
-g=dba]
-ou=cae_admin
-og=simulation_administrator
-or=simulation_administrator
-SMItemKeyList=item-id=0001;item-id=0002;item-id=0003
-SMRevList =A;A;C
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-ou

(Optional) Specifies the owning user ID.

This user owns all created objects.

-og

(Optional) Specifies the group associated with the owning user.

Note:

If you specify the **-ou** argument and do not specify the **-og** argument, the user's default group is used.

-or

(Optional) Specifies the role associated with the owning user.

-SMItemKeyList

Specifies the list of structure map items key. At least one **SMItemKey** is required, for example, **-SMItemKey=item-id=value**.

When using multiple item keys, use a semicolon separated string, for example, **SMItemKeyList=item1key;item2key;item3key**.

-SMRevList

Specifies the list of structure map item revision IDs for corresponding item keys.

Note:

This list must have the same number of entries as the **SMItemKeyList** list.

When using multiple item keys, use a semicolon separated string, for example, **SMRevList=item1-revision-id;item2-revision2-id;item3-revision-id**.

-h

Displays help for this utility.

EXAMPLES

To validate the structure map, enter the following command in a single line:

```
cae_validate_structuremap
-u=CAE_analyst
-p=*****
-g=CAE_designer
-ou=cae_admin
```

```
-og=simulation_administrator  
-or=simulation_administrator  
SMItemKeyList=item-id=0001;item-id=0002;item-id=0003  
-SMRevList=A;A;C
```




19. Teamcenter Mechatronics Process Management utilities

associate_domain_data

Adds or removes domain information from design artifacts.

SYNTAX

```
associate_domain_data [-u=user-id {-p=password |  
-pf=password-file} -g=group]  
[-action=add/remove][-domain=domain-name][-inputfile=full-path-to-input-file]  
[-objectsToUpdate=list-of-objectuid]
```

ARGUMENTS

-u

Specifies the user ID.

The user must be either a DBA user or one with write access to the design artifacts for which they want to associate domain data.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

Add means assign the domain information to the specified objects. Remove means delete the domain information associated with the specified objects.

-domain

Specifies the domain name to be associated to the specified set of input objects.

-inputfile

Specifies the full path of the file containing the list of items to be associated with the domain information.

-objectsToUpdate

Specifies the items to be associated with the specified domain information.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To associate mechanical domain with objects specified in the **uids.txt** input file, enter the following command on a single line:

```
associate_domain_data -u=Tc-admin-user -p=password -g=group -action=add  
-domain="Mechanical" -inputfile="C:\uids.txt"
```

configure_mdo_subscriptions

Creates new subscriptions or deletes existing subscriptions for MDO notifications.

Notifications are available by subscriptions only. You must subscribe to the events for which you want to get notifications. You can subscribe to the notifications by object type and event type.

SYNTAX

```
configure_mdo_subscriptions [-u=user-id {-p=password |  
-pf=password-file} -g=group]  
[-action=ON/OFF][-targettype=CLASSNAME]  
[-eventtypesEVENTNAME][-logfile]
```

ARGUMENTS

-u

Specifies the user ID.

Only the user who is part of the **MDONotificationAdministartion** authorization rule can use this utility to configure subscriptions. This is generally a DBA user or another user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-action

ON creates class-based subscription for the specified class and the event types. *OFF* removes class-based subscription for the specified class and event types.

-targettype

Specifies the class name for which the subscription must be turned on/off.

-eventtypes

Specifies comma-separated event types such as *event1,event2*. The class subscription is created or removed for the event types specified in this argument.

-logfile

Specifies the full path to the log file that records information about this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To remove the **create** event subscription for the **ModelElement** class, enter the following command on a single line:

```
configure_mdo_subscriptions -u=Tc-admin-user -p=password -g=group
-action= OFF targettype=Mdl0ModelElement -eventtypes= _Create
```

install_algebraicformulas

Provides capabilities to create algebraic formula definitions (identical, linear, quadratic, and rational) in Teamcenter.

If algebraic formulas provided in Teamcenter are deleted, you can run this utility to reinstall them.

SYNTAX

```
install_algebraicformulas [-u=user-id {-p=password |  
-pf=password-file} -g=group]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To create out of the box algebraic formula definitions in Teamcenter, enter the following command on a single line:

```
install_algebraicformulas -u=Tc-admin-user -p=password -g=group
```

install_kbl

Extends the schema to provide Teamcenter support of wire harnesses meeting the KBL standard.

Note:

This utility installs all KBL types. If any of these types already exist in the system, it is skipped and a warning is displayed in the console. The message also gets printed in the system log file.

SYNTAX

```
install_kbl [-u=Tc-admin-user {-p=password | -pf=password-file} -g=group] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

mdonotification_cleanup

Deletes MDO notifications.

SYNTAX

```
mdonotification_cleanup [-u=user-id {-p=password |
-pf=password-file} -g=group]
[-all]
[-date=]date[-mode=]Report/Execute[-objectuids=]objects
[-actionOnNotification=]create/replace/remove/revise [-logfile= ]
```

ARGUMENTS

-u

Specifies the user ID.

Only the user who is part of the **MDONotificationAdministartion** authorization rule can use this utility to clean up notifications. This is generally a DBA user or another user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-all

If this argument is used, all MDO notifications are deleted.

If **-all** is passed, other input arguments to the utility are not processed. If it is not passed, other input arguments for criteria are processed.

-date

All MDO notifications with timestamp prior to the specified date are deleted.

-mode

Report mode reports the list of MDO notifications that are identified for removal. Execute mode deletes the notifications based on specified criteria.

-objectuids

MDO notifications that have the specified object UIDs as **mdo0TriggeringComponent** are deleted.

-actionOnNotification

MDO notifications generated for the specified actions are deleted. For example, if create and revise are passed, all MDO notifications generated for create and revise events are deleted.

It can be used in conjunction with the **-objectuids** input.

-logfile

Specifies the full path to the log file that records information about this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To delete notifications for all create events, enter the following command on a single line:

```
mdonotification_cleanup -u=Tc-admin-user -p=password -g=group
-actionOnNotification=create -mode=execute -logfile"C:\abc.txt"
```

- To generate a report about all notifications in a log file, enter the following command on a single line:

```
mdonotification_cleanup -u=Tc-admin-user -p=password -g=group -all  
-mode=report -logfile"C:\xyz.txt"
```


migrate_eda_data

Allows you to bulk migrate EDA data to the current data model.

Run this utility to *manually* perform a bulk EDA migration. This utility runs automatically when you upgrade from a previous database to a more recent data model and select **Teamcenter EDA Server Support** as part of the upgrade.

Note:

The migration process assumes the data does not contain variants.

SYNTAX

```
migrate_eda_data [-u=user-id {-p=password | -pf=password-file} -g=group]
[-dryrun] [-ccaSelectFile=path-name] [-migrationList=path-name]
[-logFile=path-name] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-dryrun

Performs a dry run of the migration, making no changes to the database. Use this argument to identify problematic data prior to migration.

-ccaSelectFile

Specifies the input file containing the list of items to select when a schematic is identified as being related to multiple CCA item objects. The utility uses this information to migrate EDA schematic data to one of x number of CCA item objects related to it.

The file format is:

```
Schematic-Item-ID, CCA-Item-ID
Schematic-Item-ID2, CCA-Item-ID2
...
```

Note:

Inconsistencies in the file are logged as errors. Erroneous entries are skipped during migration.

-migrationList

Specifies the input file containing the list of items to migrate. If this argument is not provided, all **EDASchem** item objects are migrated.

The file format is:

```
Schematic-Item-ID
Schematic-Item-ID2
Schematic-Item-ID3
...
```

Note:

Inconsistencies in the file are logged as errors. Possible errors include, but are not limited to: item IDs of non-**EDASchem** item objects, duplicate entries, and invalid item IDs. Erroneous entries are skipped during migration.

-logFile

Specifies the location of the migration log file. If this argument is not specified, the default location is *TEMP\program-name_date-time.log*.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

The log file is generated in the user's **TEMP** directory. The file name format is **migrate_eda_data-date-time.log**.

RESTRICTIONS

None.

EXAMPLES

To manually perform a bulk migrate of EDA data from a previous database to a more recent data model:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group
```

To perform a dry run of the EDA data bulk migration:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group -dryrun
```

To perform a dry run of the EDA data bulk migration using the **selectionFile.txt** CCA selection file:

```
migrate_eda_data -dryrun -u=Tc-admin-user -p=password -g=group  
-ccaSelectFile=D:\migration\selectionFile.txt
```

To perform a dry run of the EDA data bulk migration using the **selectionFile.txt** CCA selection file and the **edaSchemList.txt** migration list:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group -dryrun  
-ccaSelectFile=D:\migration\selectionFile.txt  
-migrationList=D:\migration\edaSchemList.txt
```

To bulk migrate the EDA data specified in the **selectionFile.txt** migration list:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group  
-ccaSelectFile=D:\migration\selectionFile.txt  
-migrationList=D:\migration\edaSchemList.txt
```

To bulk migrate the EDA data specified in the **selectionFile.txt** CCA selection file:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group-  
ccaSelectFile=D:\migration\selectionFile.txt
```

To bulk migrate the EDA data specified in the **selectionFile.txt** CCA selection file, with the migration information sent to the **migration_output.log** log file:

```
migrate_eda_data -u=Tc-admin-user -p=password -g=group  
-ccaSelectFile=D:\migration\selectionFile.txt  
-logFile=D:\migration\migration_output.log
```

update_gde_types

Allows site administrators to update the parent types of existing GDE types based on information provided in an input file.

SYNTAX

```
update_gde_types [-u=user-id {-p=password | -pf=password-file} -g=group]
[-s=parent-type -t=type1,type2] | -f=input-file -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the input file containing one or more lines with parent type and child type GDE information in the following format:

parent_type child_type_name1, child_type_name2, child_type_name3, . . .

If the input file is specified, it takes precedence over information provided by the **-s** and **-t** arguments.

-s

Specifies the parent type to be set for the GDE types.

-t

Specifies the GDE types, separated by commas, of the parent type to be updated.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- Enter the following command on a single line to update specific children of a GDE type:

```
update_gde_types -u=user-name -p=password -g=dba -s=InterfaceDefinition
-t=port1,port2
```

- Enter the following command on a single line to update GDE types based on an input file:

```
update_gde_types -u=user-name -p=password -g=dba -f=test.txt
```

The following is an example of format of the input file:

```
#Parent child1,child2
InterfaceDefinition port1,port2,
ProcessVariable pv1,pv2,
```



20. Volume and database management utilities

collect_garbage

Collects unreferenced workspace objects and places them in a **WASTE BASKET** folder in the **Home** folder of the user. Datasets, envelopes, folders, items, and forms objects can be collected. Released objects are not collected. A special case operation is the orphan option that collects item revisions that do not have valid parent items. These items revisions may be referenced in other folders, in which case you are warned while collection is taking place. Orphan operations should not be combined with other operations.

The **collect_garbage** utility should be run in two phases. The first phase is run without the **-delete** option, which allows objects to be collected in the **WASTE BASKET** folder. This allows you to examine the contents of the waste basket. Once you are satisfied with the contents, the **collect_garbage** can be rerun with the **-delete** option to empty the **WASTE BASKET** folder.

When working with large databases, use the **-query** argument to create a report of all unreferenced objects. You can restrict the report to specific object types with various arguments. When working with a large report, you can split the report into separate files that can be executed in batches. Use the **-rf** and **-if** arguments to define the file to which you want to write the report. The batch jobs can then be executed simultaneously on multiple workstations. Use the **-delete** argument to delete the unreferenced objects from the specified folder.

SYNTAX

```
collect_garbage [-u=user-id {-p=password | -pf=password-file} -g=group]
-rf=report-file-name -if=input-file-name
[-dataset] [-item] [-occurrence] [-absoccdatalqualifier]
[-form] [-folder] [-envelope] [-all]
[-orphan] [-delete] [-report] [-query]
[-dataset] [-child_references] [-ignore_relation]
[-gsidentity] [-plmappuid=ReportOnly | ReportAndDelete]
[-start=number] [-end=number] [-h]
```

Caution:

You must run Teamcenter Workspace with system administration privileges to access the **WASTE BASKET** folder.

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-dataset

Specifies that datasets be collected or deleted.

-item

Specifies that items be collected or deleted.

Note:

This argument also collects all item revisions associated with the item.

-occurrence

Specifies that occurrences (appearance path nodes, absolute occurrences, and occurrence threads) are collected.

-absoccddataqualifier

Collects or deletes locally owned unreferenced, and locally owned referenced, absolute occurrence data qualifier (**AbsOccDataQualifier**) objects associated with replica BOM view revisions (BVRs).

-form

Specifies that forms be collected or deleted.

-folder

Specifies that object folders be collected or deleted.

-envelope

Specifies that envelopes be collected or deleted.

-all

Collects or deletes datasets, items, forms, object folders, and envelopes. This argument does not include orphans.

Siemens Digital Industries Software does not recommend using this argument when processing a large database.

-orphan

Collects or deletes all item revisions that do not have a valid parent item. These item revisions may be referenced in other folders, in which case you are warned while collection is taking place. Use the **-orphan** argument alone to collect orphan item revisions in the **WASTE BASKET** folder. Use the **-delete** argument in conjunction with the **-orphan** argument to delete orphans from the **WASTE BASKET** folder. Orphan operations should not be combined with other operations. See restriction #2.

Note:

Because orphan operations collect or delete item revisions that do not have valid parent items, it is normal for the **collect_garbage** application log file to include some errors. In most cases, you can disregard them.

-delete

Deletes all objects of a specified type. One or more of the **-dataset**, **-item**, **-form**, **-folder**, or **-envelope** arguments or the **-all** or **-orphan** arguments must be supplied. You can also include the **-gsidentity** option with the **-delete** argument.

-query

Queries the database for the instances of the specified object type when used in combination with a defined object type.

Note:

This argument works only in combination with an object type argument (**-item**, **-dataset**, **-form**, **-envelope**, **-folder**, **-all**) and the **-rf** argument.

-report

Creates a report of the objects moved to the **WASTE BASKET** folder of the user.

Note:

The **-report** argument generates output in a different format than the **-orphan** operations because orphan objects are not valid workspace objects.

-gsidentity

Locates and removes all invalid **GSIdentity** records for objects that do not exist.

-plmappuid

Deletes unreferenced entries from the **plmappuid** table.

When objects are imported into Teamcenter using PLM XML, an **Application Ref** tag can be used to define individual IDs for the imported objects. The IDs are stored in the **plmappuid** table. If these objects are later deleted in Teamcenter, the entries are not deleted from the table. Over time the table size increases, decreasing performance.

Use the **ReportOnly** value to generate a count of all the unreferenced entries in the **plmappuid** table. Use the **ReportAndDelete** value to generate a count of all the unreferenced entries in the table and delete them.

-rf= *report-file-name*

Creates a report file listing instances of a specified type. This argument may be used in combination with object type arguments (**-item**, **-dataset**, **-form**, **-envelope**, **-folder**, **-orphan**) and the **-query** argument.

When used in combination with object type arguments, the **-rf** argument retrieves a list of all instances of a specified class and writes the list to a specified file.

Note:

This argument works only in combination with an object type argument (**-item**, **-dataset**, **-form**, **-envelope**, **-folder**, **-all**) and the **-if** argument.

A file name is required when using this argument. If a file name is not provided, the list is written to a default file named *argument-name_report.txt*, where argument name is equal to **item**, **dataset**, **form**, **folder**, **envelope**, or **orphan**. If the default file already exists in the directory where this utility is executed, instances are overwritten to the default file.

When the report is large, Siemens Digital Industries Software recommends that it be split into multiple reports. The suggested naming convention is *argument-name_report_aa.txt*, *argument-name_report_ab.txt*, and so on.

-if= *input-file-name*

Uses the report file name as input to identify the unreferenced objects of a given object type. Unreferenced objects are placed in a specified subfolder within the **Waste Basket** folder.

Note:

This argument works only in combination with an object type argument (**-item**, **-dataset**, **-form**, **-envelope**, **-folder**, **-all**) and the **-if** argument. If no value is specified for this argument, the utility exits with a message.

-start

Specifies the starting number of objects to process. The default value is **1**. Use this option in conjunction with the **-end** option.

The **-start** and **-end** arguments are recommended when the utility runs out of memory when loading too many objects of the given class for processing.

-end

Specifies the ending number of objects to process. Use this option in conjunction with the **-start** option.

-dataset

Specifies that datasets qualify as garbage for collection.

-child_references

Specifies that datasets qualify as garbage if they are unreferenced in the system but have secondary objects. Use with the **-dataset** option.

-ignore_relation

Specifies that datasets are excluded from garbage collection if they have at least one secondary object attached with any of the relations in the list. The valid value for this argument is a comma-separated list containing internal relation names. Use with the **-dataset** option.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. The **collect_garbage** utility must be run by a user with Teamcenter administration privileges. This automatically enables the bypass feature and collects and deletes all garbage objects regardless of owning user and group.

2. Do not use the **-orphan** argument with an object type argument **-dataset**, **-item**, **-form**, **-folder**, **-envelope** or **-all**.

EXAMPLES

The following examples illustrate how to use the **-query** argument with this utility:

- The following example displays a message and exits the program because no file name was provided for the **-rf** argument:

```
collect_garbage -u=Tc-admin-user -p=password -g=group -item -rf -query
```

- The following example collects a list of unreferenced objects of the type **item** and writes the report to the **list_items.txt** file:

```
collect_garbage -u=Tc-admin-user -p=password -g=group -item  
-rf=list_items.txt  
-query
```

- The following example collects a list of unreferenced objects of type **item** and writes the report to the **item_report.txt** file:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -query
```

- To collect unreferenced folders, enter the following command on a single line:

```
collect_garbage -folder
```

- To collect unreferenced folders and items, enter the following command on a single line:

```
collect_garbage -folder -item
```

- To collect unreferenced folders and items and get a report to **stdout**, enter the following command on a single line:

```
collect_garbage -folder -item -report
```

- To collect all unreferenced objects except orphans, enter the following command on a single line:

```
collect_garbage -all
```

- To delete folders collected in the **WASTE BASKET** folder, enter the following command on a single line:

```
collect_garbage -folder -delete
```

- To delete all objects collected in the **WASTE BASKET** folder except orphans, enter the following command on a single line:

```
collect_garbage -all -delete
```

- To collect all item revisions with no parent item, enter the following command on a single line:

```
collect_garbage -orphan
```

- To delete orphan item revisions in the **WASTE BASKET**, enter the following command on a single line:

```
collect_garbage -orphan -delete
```

- To collect unreferenced items into the **item_rep** file, enter the following command on a single line:

```
collect_garbage -item -query -rf=item_rep
```

- To process these items and insert into folder, enter the following command on a single line:

```
collect_garbage -item -report -if=item_rep
```

- To delete items in the **SUB WASTE BASKET** folder, enter the following command on a single line:

```
collect_garbage -item -delete -sub_folder=WBITEM_item_rep
```

- To collect unreferenced forms when there are too many forms in the database to load in memory, enter the following command on a single line:

```
collect_garbage -form -end=1000
collect_garbage -form -delete
collect_garbage -form -end=1000
collect_garbage -form -delete
```

Or

```
collect_garbage -form -start=1 -end=1000
collect_garbage -form -start=1001 -end=2000
collect_garbage -form -delete
```

- To delete unreferenced occurrence threads, appearance path nodes, and absolute occurrences, enter the following command on a single line:

```
collect_garbage -occurrence -delete
```

- To collect all locally owned referenced, and locally owned unreferenced **AbsOccDataQualifier** objects and place them in the respective **BadAbsOccFolder** and **BadAbsOccFolder_unrefdirectory** folders:

```
collect_garbage -absoccdtaqualifier -query -sub_folder=
BadAbsOccFolder
```

- To delete all locally owned unreferenced **AbsOccDataQualifier** objects:

```
collect_garbage -absoccdtaqualifier -delete
```

- To delete all locally owned referenced **AbsOccDataQualifier** objects:

```
collect_garbage -absoccdtaqualifier -delete -sub_folder=
BadAbsOccFolder
```

When working with a large database, or with a large number of instances in a report file, Siemens Digital Industries Software recommends that you split the query report into multiple files. The following examples illustrate how to split a report into specified files:

- The following example splits 50,000 instances reported from a query into files of 5,000 lines each:

```
split -l 5000 item_rep.txt ITEM_rep_
```

- The following example processes each instance from the report and identifies unreferenced objects. These objects are placed in a subfolder of the **WASTE BASKET** folder:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item
-if=list_items.txt
```

- The following example processes each instance from the report and identifies unreferenced objects. These objects are placed in a subfolder of the **WASTE BASKET** folder. It also displays the object information, such as object name, ID, object type and owner's name.

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item
-if=list_items.txt
-report
```

- The following example retrieves all unreferenced objects of type **item** and places them in the **WASTE BASKET** folder:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item
```

- The following example retrieves all unreferenced objects of type **item** and places them in the **WASTE BASKET** folder. It also displays object information, such as object name, ID, object type, and owner's name.

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -report
```

The following examples illustrate how to use the **-delete** argument to delete unreferenced objects:

- The following example deletes all objects in the **WBITEM_item_rep.txt** subfolder within the **WASTE BASKET** folder.

```
collect_garbage -item -delete -sub_folder=WBITEM_item_rep.txt
```

- The following example deletes all unreferenced instances of type **item** in the **WASTE BASKET** folder, as well as all instances from any subfolders with names beginning with **WBITEM_**:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -delete
```

- The following example displays a message and exits the program because no subfolder value is specified:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -sub_folder=
```

- The following example deletes all objects of type **folder**, but does not delete the contents of the folder object. (The same concept is true for other object types, such as item, dataset, form, and envelope.)

```
collect_garbage -u=admin-user -p=admin-password -g=dba -folder -delete
```

- The following example deletes all **GSIdentity** records from the database:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -gsidentity  
-delete -rf=report-file-name
```

The following examples illustrate how to use the **-rf** argument:

- The following example displays a message and exits the program because no file name is provided for the **-rf** argument:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -rf -query
```

- The following example collects a list of unreleased objects of type **item** and writes them to the **list_items.txt** file:


```
collect_garbage -u=admin-user -p=admin-password -g=dba -item
-rf=list_items.txt
-query
```

- The following example collects a list of unreleased objects of type **item** and writes them to the **item_report.txt** file:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item -query
```

- The following example generates a list of objects for the following types: item, dataset, form, envelope, and folder. The list is placed in the following files, respectively: **item_report.txt**, **dataset_report.txt**, **form_report.txt**, **envelope_report.txt**, and **folder_report.txt**.

```
collect_garbage -all -query
```

The following examples illustrate how to use the **-if** argument:

- The following example displays a message and exits the program because no value was provided for the **-if** argument:

```
collect_garbage -item -if=
```

- The following example checks each entry in the **item_report_aa** file. A folder named **WBITEM_item_report_aa** is created within the **WASTE BASKET** folder and all unreferenced objects are placed within this folder.

```
collect_garbage -item -if=item_report_aa
```

- The following example checks each entry in the given list and identifies unreferenced objects. A subfolder is created within the **WASTE BASKET** folder and all unreferenced objects are placed within this subfolder. It also displays object information, such as object name, ID, object type and owner's name.

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item
-if=list_items.txt
-report
```

The following examples illustrate how to use the **-sub_folder** argument:

- The following example deletes the objects of type **dataset** within the **WBDSET_dataset_report1** folder within the **WASTE BASKET** folder.

```
collect_garbage -dataset -sub_folder=WBDSET_dataset_report1 -delete
```

- The following example deletes all instances of type **dataset** within the **WASTE BASKET** folder and all instances in the subfolder named **WBDSET_file-name**:

```
collect_garbage -dataset -delete
```

This section illustrates how to use the **-query** argument. The following example compiles a list of objects of type **item** that are not released and writes the list to the **list_items.txt** file:

```
collect_garbage -u=admin-user -p=admin-password -g=dba -item  
-rf=list_items.txt  
-query
```

The following example shows how to use the **-dataset**, **-child_references**, and **-ignore_relation** arguments. In the example, all unreferenced datasets that have secondary objects are collected as garbage, except those that have any of the secondary object attached with the **IMAN_Rendering** or **IMAN_specification** relationship.

```
collect_garbage -u=admin-user -p=<password> -g=dba  
-dataset -query -rf=report.txt -child_references  
-ignore_relation=IMAN_Rendering,IMAN_specification
```

IMPORTANT NOTES

- The **-report** option for orphan operations outputs in a different format than for other object types, because orphans are not valid workspace objects.
- The **-item** option collects the associated item revisions along with the item.
- Errors are reported in the **logfile** when running in orphan collection mode. The error reported indicates that an error attempting to load an indirected object. These errors can be disregarded.

dataset_cleanup

Repairs corrupted datasets and removes orphaned datasets and revision anchors.

Caution:

Siemens Digital Industries Software recommends that you run this utility only when there is no other activity on the database.

PROBLEM IDENTIFIERS

A dataset is identified as corrupted if any of the following problems are found:

- Dataset has no reference to an **ImanFile** object.
- Dataset has reference to an **ImanFile** object, but the corresponding operating system file does not exist and the dataset is not archived.
- Dataset is an orphan (that is, the dataset refers to the anchor but the anchor does not go to dataset).
- Anchor refers to datasets that do not exist.
- Anchor size = 0.

OBJECT CLEANUP RULES

A dataset object is reattached to revision anchor if it is an orphan but is referenced by some other objects, or deleted if it meets the following criteria:

- Dataset is an orphan and is not referenced.
- Dataset is not archived and the associated operating system file does not exist.

ANCHOR CLEANUP RULES

The **dataset_cleanup** utility repairs dataset revision anchors as follows:

- If the anchor refers to nonexistent datasets, the references are removed from the anchor.
- If the anchor size = 0, the anchor is deleted.

SYNTAX

```
dataset_cleanup [-u=user-id {-p=password | -pf=password-file} -g=group]
[-reportCorruption | -fixCorruption | -reportBadAnchor | -fixBadAnchor | -reportRestorable |
-fixRestorable | -reportOrphan | -deleteOrphan] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-reportCorruption

Generates a report of all corrupted dataset versions and anchors.

The report file, *dataset_cleanup_actions.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-fixCorruption

Either fixes or removes all corrupted dataset versions and anchors.

A report file of corrections, *dataset_cleanup.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-reportBadAnchor

Generates a report of all bad anchors.

The report file, *dataset_cleanup.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-fixBadAnchor

Fixes all bad dataset anchors.

The report file, *dataset_cleanup_fix_anchor.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-reportRestorable

Generates a report of all restorable orphaned datasets.

The report file, *dataset_cleanup.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-fixRestorable

Fixes all restorable orphaned datasets.

The report file, *dataset_cleanup_restore.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-reportOrphan

Generates a report of all invalid and orphaned datasets.

The report file, *dataset_cleanup.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-deleteOrphan

Deletes all invalid and orphaned datasets.

The report file, *dataset_cleanup_delete.log*, is generated in the running directory and overwrites any existing *dataset_cleanup.log* file.

-h

Displays help for this utility.

LEGACY

dataset_cleanup [-u=user-id {-p=password | -pf=password-file} -g=group]
 -rf=file-name | -if=file-name
 [-of=log-file-name] [-b=beginning-anchor]
 [-e=ending-anchor] [-start_date=start-date] [-end_date=end-date] -h

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-rf

Creates a report file listing the corrupted datasets. If the location of the output file should be other than the current folder, include the path as well as the output file name.

-if

Uses the report file as input to purge corrupted and orphaned datasets and repair revision anchors.

Caution:

This option deletes orphaned datasets. Carefully examine the input file and remove from the list any orphaned datasets that you wish to retain. It is also an administrative best practice to paste important datasets into a folder to prevent inadvertent deletion.

-of

Cleans up and logs the results to a log file. This argument must be supplied if the **-if** argument is used but is optional with the **-rf** argument.

-a

Specifies that corrupt anchors (those that are orphaned and are not referenced by a dataset) be deleted and a message be provided.

-b

Specifies the first revision anchor of a contiguous series to be repaired. The default value is **1**.

-e

Specifies the last revision anchor of a contiguous series to be repaired. The default value is **last**.

A revision anchor is an object that keeps track of a set of revisions of some object. One such class of objects is datasets.

-start_date

Specifies the starting date to search for datasets that have been modified from this date. Use this argument with the **-end_date** argument.

The format of the date is *"DD-MMM-YYYY HH:MM:SS"* and must be inside the double quotes because of the space between the year and the hour. This argument is used only with the **-rf** argument.

-end_date

Specifies the ending date to search for datasets that have been modified until this date. This argument is optional and is used only with the **-start_date** argument. If this argument is not specified, the end date is the current date.

The format of the date is *"DD-MMM-YYYY HH:MM:SS"* and must be inside the double quotes because of the space between the year and the hour. This argument is used only with the **-rf** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment..

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

None.

EXAMPLES

- To generate a report file called **myreportfile** listing corrupted dataset objects, enter the following command on a single line:

```
$TC_ROOT/bin/dataset_cleanup -u=admin-user -p=admin-password -g=dba
-rf=myreportfile
```

- To run the **dataset_cleanup** utility using the **myreportfile** file as input, enter the following command on a single line:

```
$TC_ROOT/bin/dataset_cleanup -u=admin-user -p=admin-password -g=dba
-if=myreportfile -of=mylogfile
```

- On a database with 1000 dataset revision anchors, you could run the **dataset_cleanup** utility as follows:

```
$TC_BIN/dataset_cleanup -u=admin-user -p=admin-password -g=dba -b=1
-e=500
-rf=dataset_cleanup_500.report
$TC_BIN/dataset_cleanup -u=admin-user -p=admin-password -g=dba
-b=501 -e=1000 -rf=dataset_cleanup_1000.report
```

- To purge all datasets with modification dates between Oct-01-2020 and Oct-10-2020:

```
dataset_cleanup -u=admin-user -p=admin-password -g=dba
-start_date="01-OCT-2020 00:00:00" -end_date="10-Oct-2020 00:00:00"
-rf=ttt.txt
```

- To purge all datasets with modification dates from Oct-01-2020 to the current date:

```
dataset_cleanup -u=admin-user -p=admin-password -g=dba
-start_date="01-OCT-2020 00:00:00" -rf=ttt.txt
```

CLEANING UP DATASETS AND REPAIRING REVISION ANCHORS

Perform the following steps to clean up corrupted datasets:

1. Use the **dataset_cleanup** utility to generate a report file called **myreportfile** listing the corrupted dataset objects in the database by entering the following command on a single line:

```
$TC_ROOT/bin/dataset_cleanup -u=admin-user -p=admin-password -g=dba
-rf=myreportfile
```

The report file contains a list of corrupted datasets sorted by **Object_UID**. The report also contains the problem identifier, dataset name, and ownership.

If the **-a** argument is specified on the command line, the utility deletes the corrupt anchors and displays a message to the user. If the **-a** argument is not supplied, a message is displayed indicating that the anchor was skipped and the **-a** option should be used.

You must review the report file and decide which datasets, if any, should not be purged from the database.

2. Use a text editor to remove any references to dataset objects that should not be purged from the database from the report file.
3. Run the **dataset_cleanup** utility using the **myreportfile** file as input to purge corrupted dataset objects from the database or fix anchors and log the results to the **mylogfile** file by entering the following command on a single line:

```
$TC_ROOT/bin/dataset_cleanup -u=admin-user -p=password -g=dba
-if=myreportfile -of=mylogfile
```

The utility attempts to fix the revision anchor, attach the dataset to another revision anchor, or purge the datasets from the database.

A final output report is generated showing the results for each dataset. The report displays the following message if the operation is successful:

```
problem deleted
```

If the operation is unsuccessful, the following message is displayed:

```
could not delete error stack number
```

4. Display information about the dataset cleanup process by entering the following command:

```
ps -ef | grep data
```

5. Kill the dataset cleanup process by entering the following command:

```
kill -9 PID
```

PID is the operating system process ID returned in step 4.

delete_item_data

Deletes unused item revisions from the database. The item revisions to be removed are contained in an input file created by the user.

SYNTAX

```
delete_item_data [-u=user-id {-p=password | -pf=password-file} -g=group]
[-inputFile=input-file | -inputKeyFile=input-file]
-configFile=configuration-file [-outputDir=output-file-directory]
{-mode=report | delete} [-delimiter=delimiter] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputFile

Specifies the absolute path of the input file. The input file has the following format (the forward slash, /, is the delimiter):

```

item id/Item Revision1
item id/Item Revision2
item id/Item Revision3

```

-inputKeyFile

Specifies the absolute path of the input file. The input file has the following format:

```

[keyAttr1=keyVal1] [,keyAttr2=keyVal2]...[,keyAttrN=keyValN],
rev_id=A item_id=000100,rev_id=B

```

-configFile

Specifies the absolute path of the configuration file. The configuration file has the following field names and format:

```

EXCLUDE = Relationship@C=Class; Relationship@T=Type; ...;
INCLUDE = Relationship@T=Type; Relationship@C=Class; ...;

```

The **EXCLUDE** and **INCLUDE** expressions contain three values as shown in the following examples:

```

Relationship@C=Class
Relationship@T=Type

```

- The first value of the expression contains the relation value followed by the @ separator.
- The second value of the expression contains the type name (T) or the class name (C) followed by the = separator.
- The third value of the expression contains the value of the type or class followed by the ; delimiter.

Note:

- If there is only one relation specified in the expression, it must be terminated with the ; delimiter. For example:

```
IMAN_reference;
```

If there is more than one expression, the expressions must be separated by the ; separator. For example:

```
IMAN_specification@T=MSWord; IMAN_reference@C=
PSBOMViewRevision;
```

- To exclude the based-on item revisions to be deleted, add the **IMAN_based_on@C=ItemRevision;** expression to the **EXCLUDE** section.

For example, assume item revision B was revised from A. When item revision B is deleted, revision A will also be deleted unless the **IMAN_based_on@C=ItemRevision;** expression is in the **EXCLUDE** section.

This argument is required.

-outputDir

Specifies the path where report and log files are to be written.

The default value is the current directory.

-mode

Specifies one of the following the modes:

- **report**

Generates a summary report containing the following information:

- Target item revision ID
- Reference status
- Usage status
- Site ownership
- Exported replica status
- Deletion status

- **delete**

Deletes the item revision and its associated objects except those specifically excluded by entries in the configuration file. To be deleted, the associated objects must only be related to the item revision. If the item revision is the only revision of an item, the item is also deleted. If the object is a dataset, all versions of the target dataset are deleted as well as all forms and named references associated with the dataset.

This argument is required.

-delimiter

Specifies the delimiter character separator between the item ID and the item revision ID. The default is the forward slash (/).

-h

Displays help for this utility.

RESTRICTIONS

None.

EXAMPLES

- The following is an example of an input file:

```
ABC000075/A
ABC000074/A
ABC000092/A
ABN000002/A
ABN000011/A
ABN000058/A
```

- The following is an example of a configuration file:

```
EXCLUDE = IMAN_specification@T=UGMASTER;
INCLUDE = IMAN_specification@T=MSWord;IMAN_reference@C=

PSBOMViewRevision;
```

If there are no configuration entries in the configuration file, the **EXCLUDE** and **INCLUDE** values must be assigned with the ; delimiter as shown below:

```
EXCLUDE=;
INCLUDE=;
```

Any statements not conforming to the format above are not processed for evaluation. This example indicates that the attached objects are not processed for evaluation and but they are only dereferenced from their parent item revision.

- The following is an example of the **delete_item_data** command line entry:

```
delete_item_data -u=admin-user -p=admin-password -g=dba
-inputFile=c:\temp\input.txt
-configFile=c:\temp\config.txt -mode=report
```

- The following is an example of using the **delimiter=@** argument:

```
000001@A
000002@B
```

hsm_capacity_alert

Evaluates if the Teamcenter volume tiers filled capacity exceeds the specified alert capacity levels. When the filled capacity exceeds the alert capacity level, an e-mail is sent to the system administrator. The capacity levels are measured in percentage of total capacity.

Sites can schedule this utility to execute overnight using a Linux **cron** job or the Microsoft Windows **at** command. This utility can require considerable time to evaluate the capacity levels on different tiers.

SYNTAX

```
hsm_capacity_alert [-u=user-id {-p=password | -pf=password-file} -g=group]
-alertcapacity=percentage -tier=tier-level [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-alertcapacity

Specifies alert capacity as a percentage of total capacity. You can include the percent symbol (%) in this argument.

-tier

Specifies volume tier. Valid values are **1**, primary, and **2** secondary.

-v

Verbose mode. Provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility requires that the **HSM_primary_tier_hosts** and **HSM_secondary_tier_capacity** preferences contain estimated total capacity values. These preferences are set using **preferences_manager** utility.

RETURN VALUES

Return value 0
upon
success

Return value 1
upon failure

EXAMPLES

To determine if the primary tier capacity exceeds 80% of total capacity, enter the following command:

```
hsm_capacity_alert -u=admin-user -p=admin-password -g=dba -tier=1
-alertcapacity=80%
```

When the filled capacity exceeds 80 percent of total capacity level, an e-mail is sent to the system administrator.

hsm_report

Evaluates any or all hierarchical storage management (HSM) policies to generate a report. Because of performance considerations using the rich client application interface, Siemens Digital Industries Software recommends the system administrator execute this utility to create pending migration file sets.

Sites can schedule this utility to execute overnight through a Linux **cron** job or Microsoft Windows **at** command. This utility can take considerable time to evaluate all migration policies.

SYNTAX

```
hsm_report [-u=user-id {-p=password | -pf=password-file} -g=group]
-tier={1 | 2 | 3} -migrationreport=[ALL |migration-policy] [-before=before-date]
[-after=after-date] -filepath=filepath
[-listpolicies=ALL | policy-name] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-tier

Specifies migration tier. Valid values are:

1	Primary to secondary.
2	Secondary to tertiary.
3	Primary to tertiary.

-migrationreport

Specifies migration policy for which the report need to be reported. To generate a report for all active migration policies, specify **-migrationreport=ALL**.

-before

Specifies the beginning date for reporting migration, for example, **23_Mar_2004**.

-after

Specifies the ending date for reporting migration, for example, **23_May_2006**.

-filepath

Specifies the operating system file path to which the report is saved.

-listpolicies

Indicates that the utility is to list all active policies.

-v

Verbose mode provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None

EXAMPLES

- Execute the following command to generate report on all policies in the database:

```
hsm_report -u=admin-user -p=admin-password -g=dba  
-migrationreport=ALL -tier=1 -filepath=c:\temp\report.txt
```

- Execute the following command to generate report about a specific policy in the database:

```
hsm_report -u=admin-user -p=admin-password -g=dba  
-migrationreport=policy-name -filepath=c:\temp\report.txt
```

- Execute the following command to list of all active policies names defined for a database:

```
hsm_report -u=admin-user -p=admin-password -g=dba -listpolicies
```

index_verifier

Detects missing indexes in a Teamcenter database and describes how to create the indexes. This utility ensures that all indexes created at installation or upgrade are in place by consulting the internal Teamcenter **pom_indices** data dictionary. The **index_verifier** utility examines each Teamcenter object class and checks for missing indexes. If it finds missing indexes, it reports them in the output file.

Use the **index_verifier** utility with the **-o=DO_IT** argument to find the missing indexes and create the replacement indexes in one step, for example:

```
index_verifier -u=username -p=password -g=group -o=DO_IT
```

There are five types of indexes that can be detected using this utility:

- Indexes on the primary key of each Teamcenter class.

The PUID (internal attribute of each table mapped to a Teamcenter class) must have a unique index.

- Indexes on variable length arrays (VLA).

Each VLA must have two indexes, as follows:

- An index on the PUID+PSEQ. This index must be unique.
- An index on the PVAL attribute (**pvalu_0** for **POM_typed/untyped_reference** and **pval_0** for the other data type).
- Indexes created by Teamcenter.

These indexes are created using Teamcenter POM ITK and information about these indexes resides in the POM data dictionary **pom_indexes** table.

- Functional indexes

This utility detects the necessary functional indexes required by the version of Teamcenter in use. These functional indexes may include, but are not limited to the following:

- A functional index on **WorkspaceObject.object_type**
- A functional index on **WorkspaceObject.object_desc**
- A functional index on **WorkspaceObject.object_name**
- A functional index on **Item.Item_id**
- A functional index on **ItemRevision.Item_revision_id**

- Indexes on system tables such as **pom_backpointer**, **pom_m_lock**, and the **pm_process_list** tables.

SYNTAX

```
index_verifier [-u=user-id {-p=password | -pf=password-file} -g=group]
[-o= [DRYRUN|DO_IT]] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument. One of the two mutually exclusive password elements is required.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-o

Specifies the action to take with the output of the **index_verifier** utility. The **-o** argument has the following parameters.

-o DRYRUN

Specifies to identify missing indexes but *not* create the missing indexes in the database. This is the same behavior as not using the **-o** argument.

-o DO_IT

Specifies to identify missing indexes and immediately create missing indexes in the database.

For Oracle, if the **SA** user who runs **index_verifier** wants missing indexes to be created in a specific tablespace, the **SA** user must set the **TC_INDEX_STORAGE** environment variable:

```
TC_INDEX_STORAGE TABLESPACE tablespace-name
```

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility can be run while users are logged on to Teamcenter.

mark_for_migrate

Evaluates any or all active policies to determine if there are any Teamcenter volume files pending for migration, and if found, mark it for migration. Because of performance considerations using the rich client application interface, Siemens Digital Industries Software recommends the system administrator execute this utility to create pending migration file sets.

Sites can schedule this utility to execute overnight through a Linux **cron** job or Microsoft Windows **at** command. This utility can take considerable time to evaluate all migration policies.

SYNTAX

```
mark_for_migrate [-u=user-id {-p=password | -pf=password-file} -g=group]
-migrationpolicy=[ALL | policy-name]
[-listpolicies] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-migrationpolicy

Evaluates a given migration policy and marks pending files for migration. To evaluate all active migration policies and mark them for migration, specify **-migrationpolicy=ALL**.

-listpolicies

List all active policies in chronological order.

-v

Verbose mode. Provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

RETURN VALUES

Return value 0
upon
success

Return value 1
upon failure

EXAMPLES

- Execute the following command to evaluate all policies in the database:

```
mark_for_migrate -u=admin-user -p=admin-password -g=dba
-migrationpolicy=ALL
```

- Execute the following command to evaluate a specific policy in the database:

```
mark_for_migrate -u=admin-user -p=admin-password -g=dba
-migrationpolicy=policy_name
```

- Execute the following command to receive a list of all active policies names defined for a database:

```
mark_for_migrate -u=admin-user -p=admin-password -g=dba -listpolicies
```


move_volume_files

Moves files either of the following ways:

- From one Teamcenter volume to another over your WAN/LAN using FMS. You can specify files to move based on file date, file age, or by volume allocation rules.
- Within the same volume from the current volume subdirectory structure into multiple directories.

For more information about using volume allocation rules to reallocate volume files, see *Teamcenter Administration*.

You can run this utility as a **cron** job or as a scheduled task using **process_move_file_volumes** as a **.sh** or **.bat** script, respectively. Use this script if your scheduling tools are not running in the Teamcenter environment with the appropriate **TC_ROOT** and **TC_DATA** variables set. The script sets these variables and calls **tc_provillevars** directly before running the **move_volume_files** utility. The script accepts the same arguments as the utility. The arguments specified in the script are run by the utility.

SYNTAX

```
move_volume_files -u=user-id {-p=password | -pf=password-file} -g=group
[-listvolumes] [-distributefiles]
[-f={list | move}] [-output_file=file-name] [-srcvol=source-volume]
[-destvol=destination-volume]
[-before=access-time-ending-range] [-after=access-time-beginning-range]
[-maxage=days] [-fszlessthan=bytes] [-fszgreaterthan=bytes]
[-outrulesfile=file-name] [-rulesfile=file-name] [-excludedvollist=file-name]
[-presorted_file=file-name] [-listsaf] [-transfersaf]
[-migrate_to_batch_saf] [-retainlmd] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

-after

Any file last accessed after this date and time will be included in the list or move operation. The format of the date is the same as the configured date/time format for the current locale. For example, in the default en_US locale, '07-Dec-2020 14:53' is the correct format. This option can be used along with the **-rulesfile** option to further refine the results.

-before

Any file last accessed before this date and time will be included in the list or move operation. The format of the date is the same as the configured date/time format for the current locale. For example, in the default en_US locale, '07-Dec-2020 14:53' is the correct format. This option can be used along with the **-rulesfile** option to further refine the results.

-destvol

Specifies the name of the Teamcenter destination volume. Can not be used with the **-distributeFiles** argument.

-excludedvollist

Specifies the path and name of the file containing a list of volumes to be excluded from both listing and transfer actions. This argument is typically used to list default local volumes (store and forward volumes) to ensure the files stored in these temporary volume location are not transferred.

Use either the full path to the file, or use the partial path/file name, in which case the utility searches for the file name in the current directory.

Any number of volumes can be specified in this file. Each entry must be a valid volume name, listed on its own row in the file.

-f

Specifies whether the utility lists or moves files.

list

Lists the files to be moved in the specified source volume that are candidates to be moved.

This argument must be used with the **-srcvol** argument.

move

Moves the Teamcenter files to the specified destination volume.

This argument must be used with the **-srcvol**, **-destvol** and **-distributefiles** arguments.

-fszlessthan

Specifies the maximum file size (in bytes) a file can reach and still be eligible for transfer. For example, if set to **64000**, all files that are 64,000 bytes or smaller can be transferred.

If the **-fszgreaterthan** argument is specified, the files selected are within that range, inclusive.

-fszgreaterthan

Specifies the minimum file size (in bytes) a file can reach and still be eligible for transfer. For example, if set to **2048**, all files that are 2,408 bytes or larger can be transferred.

If the **-fszlessthan** argument is specified, the files selected are within that range, inclusive.

-listsaf

Lists files evaluated for a store and forward operation.

-listvolumes

Displays a list of all Teamcenter volumes configured in the database.

-maxage

Specifies the maximum age (in days) a file can reach and still be eligible for transfer. For example, if set to **24**, any file 24-days old or younger can be transferred.

If used outside of a rules environment, the current date minus the age specified by this argument is used as the **-after** value.

-output_file

Specifies the file to which list output is written when using the **-f=list** or **-listsf** arguments. If not specified, the **move_tcfiles_list.txt** file is created.

-outrulesfile

Outputs the volume allocation rules XML template to the current directory. The XML template is given the filename specified by this argument. For example, if set to **VolRules.xml**, an XML template of the allocation rules named **VolRules.xml** is created and stored in the current directory.

-presorted_file

Specifies the path and name of the file containing a list of files to be moved. This argument must be used with the **-f=move** argument.

Use either the full path to the file, or use the partial path/file name, in which case the utility searches for the file name in the current directory.

Any number of files can be specified in this file. The file format is the same format as the file output by the **-f=list** argument. Typically, you generate a file using this **-f=list** argument, edit the file as necessary, and then set the **-presorted_file** argument with the name of the edited file.

-distributefiles

Distributes Teamcenter volume files into multiple subdirectories if the file count exceeds the limit set by the **TC_Volume_Max_Files_Per_Dir** preference. This argument must be used with the **-f=move** argument. Use with other utility arguments to choose the files to move. Can not be used with the **-destvol** argument.

-rulesfile

Specifies the path and name of the volume allocation rules XML file to evaluate. The file is validated against the DTD, and then its rules are evaluated. Volume files are moved or listed, based on the setting of the **-f** argument.

This argument must be used with the **-f** argument. It does not work with the **-listvolumes**, **-srcvol**, **-destvol**, or **-presorted_file** arguments.

-srcvol

Specifies the name of the Teamcenter source volume.

-transfersaf

Transfers all files evaluated for a store and forward operation from all of the user's local volumes to their corresponding default volumes. It also creates and schedules a dispatcher request to clean up files in local volumes.

Note:

The **-listsaf** and **-transfersaf** arguments require installation of dispatcher scheduler, module, client, and the **store_and_forward** translator. They also require setting the **TC_Store_and_Forward** and **FMS_SAF_Batch_Transfer_Enabled** preferences to true.

For more information, see *Teamcenter Administration*.

-migrate_to_batch_saf

Migrates from the old Store and Forward to the new one. This option should be used before moving to the new (batch) Store and Forward.

-retainlmd

Prevents the last modified date from being updated on the file named reference during a file move or transfer operation. This feature is useful to prevent the appearance that the file has been updated when its contents have not changed.

- If **retainlmd** is used when moving or transferring a file, then the 'last modified date' of the named reference maintains its value and is not updated.

- If **retainlmd** is not used, then the last modified date is updated to the current date.

-v

Executes the utility in verbose mode.

RESTRICTIONS

If the volume path definition is defined in multiple locations, the definition must be exactly the same in each location. For example, if you define a volume in the Organization application using a drive/path definition format, you must use the same drive/path definition in the FMS master configuration file (you cannot use the UNC format).

Conversely, the volume can be defined in both Organization and in the master configuration file in UNC format.

EXAMPLES

- Enter the following command on a single line to generate the list of all files in Teamcenter volume **vol003**:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -f=list
-srcvol=vol003 -destvol=vol1001 -v
```

- Enter the following command on a single line to generate the list of all files in Teamcenter volume **vol003** that were last accessed in the given date range:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -f=list -srcvol=vol003
-destvol=vol1001 -after=03-Feb-2008 -before=03-Mar-2008 -v
```

The list of evaluated files is stored in the **move_tcfiles_list.txt** file, in the current directory.

- Enter the following command on a single line to move all the files listed in the **move_tcfiles_list.txt** file from volume **vol003** to volume **vol1001**:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -f=move
-presorted_file=move_tcfiles_list.txt -destvol=vol1003 -v
```

- Enter the following command on a single line to relocate all Teamcenter files from volume **vol003** to the destination volume **newvol002**:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -f=move
-srcvol=vol003 -destvol=newvol002 -v
```

- Enter the following command on a single line to relocate Teamcenter files that were last accessed on November 29, 2006, or later, from Teamcenter **vol002** to the destination volume **newvol003**:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -f=move
-srcvol=vol002 -destvol=newvol003 -after=29-Nov-2006 -v
```

- Enter the following command on a single line to generate the volume allocation rules XML template named **VolSelectionRules.xml** and stored in the current directory:

```
move_volume_files -u=admin-user -p=admin-password -g=dba
-outrulesfile=VolSelectionRules.xml
```

- Enter the following command on a single line to evaluate the volume allocation rules in the **VolSelectionRules.xml** file and store the results in the **move_tcfiles_list.txt** file in the current directory:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -rulesfile=VolSelectionRules.xml
-f=list -v
```

- Enter the following command on a single line to evaluate the volume allocation rules in the **VolSelectionRules.xml** file and move the evaluated files:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -rulesfile=VolSelectionRules.xml
-f=move -v
```

- Enter the following command on a single line to generate the list of all files evaluated for a store and forward operation:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -listsaf -v
```

- Enter the following command on a single line to transfer files evaluated for a store and forward operation from all of the users' local volumes to their corresponding default volumes and to schedule a dispatcher request to clean up files in local volumes:

```
move_volume_files -u=admin-user -p=admin-password -g=dba -transfersaf -v
```

- Enter the following command on a single line to distribute the files from the current subdirectory structure to the new subdirectory structure within the same volume as controlled by the **TC_Volume_Max_Files_Per_Dir** preference.

```
move_volume_files -u=admin-user -p=admin-password -g=dba -distributefiles -srcvol=vol003
-v
```

- Enter the following command on a single line to evaluate the volume allocation rules in the **VolSelectionRules.xml** file and distribute the evaluated files into multiple directories (within the same volume) as controlled by the **TC_Volume_Max_Files_Per_Dir** preference.

```
move_volume_files -u=admin-user -p=admin-password -g=dba -distributefiles -srcvol=vol003
-rulesfile=VolSelectionRules.xml -v
```

purge_datasets

Removes (purges) old versions of datasets from the database and outputs a list of each dataset purged, along with the owning user and group.

Note:

Normally, Teamcenter stores a fixed number of dataset versions in the database. The maximum number of datasets retained is set using the **AE_dataset_default_keep_limit** preference.

Certain conditions prevent automatic purging of old datasets. For example, when a user does not have permission to purge a dataset owned by another user, or when a group is given read/write permission but not delete permission.

Additionally, datasets are not purged when the named references in **version0** do not match the named references in the latest dataset version. Use the **-skipInconsistencyCheck** argument to bypass this named references check. Use this argument only in situations in which you know why the named references differ and are confident that purging the older dataset versions will not result in loss of needed data. Possible situations include CAD integrations in which custom code is implemented, wrong coding exists from custom ITK programs, and PLM XML import.

Before using the **-skipInconsistencyCheck** argument, run this utility without the argument and review the output for any failed purges. Investigate all datasets that did not purge, correcting problems if necessary.

For example, **version0** of a dataset contains three named references: **a.txt**, **b.txt** and **c.txt**. The latest version of the same dataset contains two named references: **a.txt** and **b.txt**. Using this utility to purge this dataset fails because of the inconsistency between the named references. The utility logs an inconsistent data message.

In this situation, you should compare the named references between the versions and resolve the inconsistency if necessary.

- Compare the named references of the two versions in the rich client by choosing **View→Named References**.
- Correct the inconsistency by copying the **c.txt** named reference from **version0** to the clipboard, then pasting it into the latest version. The named references must be in the same order for both versions. Rerunning the utility would purge this dataset.

Alternatively, checking the dataset out, making changes, and checking it back in synchronizes the named references between the versions.

If it is not possible to synchronize the named references, for example, the datasets are already released (and thus read-only), then use the **-skipInconsistencyCheck** argument to purge the datasets.

SYNTAX

```

purge_datasets [-u=user-id {-p=password | -pf=password-file} -g=group]
[-b=beginning_anchor] [-e=ending_anchor] [-k=keep-limit]
[-set] [-report] [-replica_only] [-site=site-name]
[-start_date="DD-MMM-YYYY HH:MM:SS"]
[-end_date="DD-MMM-YYYY HH:MM:SS"]
[-itemidsfile=item-id-file-name]
[-itemKeyFile=item-key-file-name]
[-grmtypesfile=relation-types-file-name]
[-anchorfile=revision-anchor-file-name]
[-includeFolderContents]
[-skipInconsistencyCheck]
[-h]

```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, then the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-b

Specifies the first version anchor (*beginning_anchor*) of a contiguous series to be purged. The default value is **1**. Version anchors are objects that keep track of a set of versions of an object. Datasets are one such class of objects.

-e

Specifies the last version anchor (*ending_anchor*) of a contiguous series to be purged. The default value is **last**. Version anchors are objects that keep track of a set of versions of an object. Datasets are one such class of objects.

-k

Specifies the keep limit for purging.

For example, if the **-k** argument is set to **-k=4**, then a dataset is purged down to four versions.

If the **-k** argument is not used, then the current version limit of the dataset is used as the purge keep limit.

-set

For the datasets to be purged, sets the version limit to the **-k** argument value.

The **-set** argument has no effect if the **-k** argument is not used.

-report

Produces output showing the datasets to be purged, but does not purge the datasets from the database.

The output lists the item IDs of datasets that need to be purged. To use these IDs in the input file needed by the **-itemidsfile** option, manually copy the item IDs into the input file.

-replica_only

Specifies that only replica datasets are purged. The **-site** argument can be used in conjunction with the **-replica_only** argument to purge only the datasets replicated from a specific site.

-site

Specifies the site from which replica datasets are purged. Valid only in conjunction with the **-replica_only** argument.

-start_date

Specifies the start date/time for which datasets are purged.

-end_date

Specifies the end date/time for which datasets are purged. This argument is optional and must be used only with the **-start_date** argument.

-itemidsfile

Specifies an input file containing a list of item IDs. Use the **-itemidsfile** argument followed by the directory path and file name.

To obtain the item IDs, run the **purge_datasets** command with the **-report** argument.

In the input file, enter one item ID per line, or you can enter multiple item IDs on one line separated by commas.

For example, a file *item_uid.txt* might contain this list:

```
000041
000051
000078
```

-itemKeyFile

Specifies an input file containing a list of item keys <attr1=value1,attr2=value2,...>. Use the **-itemKeyFile** argument followed by the directory path and file name.

In the input file, enter one item key per line, or you can enter multiple item IDs on one line separated by commas.

-grmtypesfile

Specifies an input file containing a list of specified relation types to use. Enter one relation type per line, or enter multiple relation types on one line separated by commas.

If this parameter is not specified, then all relation types are used.

This parameter is valid only with the **-itemidsfile** argument.

-anchorfile

Specifies the name of a file that contains UUIDs of revision anchors to process.

On very large databases, the number of objects processed may make this utility run slow or fail. Using an anchor file can prevent failures and increase the speed of the dataset purge process.

To generate a list of UUIDs to process, use the **item_report** utility with its **-anchorfile** argument.

-includeFolderContents

Gets all the datasets related to the Item IDs specified for purge in the input file in **-itemidsfile**. This parameter is optional and is valid only with the **-itemidsfile** argument.

-skipInconsistencyCheck

Bypasses the consistency check of named references and purges previous versions of a dataset even if the named references in **version0** of the dataset are not the same as the named references in the latest version.

If **-skipInconsistencyCheck** is not specified, then a dataset is not purged if the named references in **version0** of the dataset are not the same as the named references in the latest version.

-h

Displays help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment..

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

The **purge_datasets** utility must be run by the a user with Teamcenter administration privileges. This automatically enables the bypass feature and purges old datasets regardless of owning user and group.

EXAMPLES

- Purge the range of contiguous datasets down to the specified keep limit, and set the version limit of the current dataset to the keep limit value:

```
$TC_BIN\purge_datasets -u=Tc-admin-user -p=password -g=group -b=beginning-  
anchor -e=ending-anchor -k=keep-limit -set
```

- Purge datasets with specified item ids, and also purge their related items:

```
$TC_BIN\purge_datasets -u=Tc-admin-user -p=password -g=group  
-itemidsfile=D:\workdir\item_uid.txt -includeFolderContents
```

purge_volumes

Removes (purges) operating system files that represent deleted Teamcenter objects. The **purge_volumes** utility can be run interactively, in forced execution mode, or periodically.

During a Teamcenter session, users delete objects. However, if an object's associated files remain in the Teamcenter volume, the **purge_volumes** utility unlocks these files so they can be deleted at the operating system level.

SYNTAX

```
purge_volumes [-u=user-id {-p=password | -pf=password-file} -g=group]  
[-f] [-s=time] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies forced execution mode. When the **-f** argument is supplied, files are deleted without prompting for confirmation. When the **-f** argument is not supplied, the **purge_volumes** utility runs interactively and prompts the user before deleting each file.

-s

Specifies sleep time in seconds. After each **purge_volumes** session is complete, it is dormant for the specified time before running again. If the **-s** argument is not supplied, the **purge_volumes** only runs once.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

1. When running the **purge_volumes** utility periodically with the **-s** argument, include the **-f** argument. This disables interactive mode and ensures that the utility runs to completion.

EXAMPLES

To run the **purge_volumes** utility interactively, enter the following command on a single line:

```
$TC_ROOT/bin/purge_volumes -u=admin-user -p=admin-password -g=dba
```

The **purge_volumes** utility prompts the user before deleting each volume. When complete, the system displays the following message:

```
Purge_Volumes: terminating
Stop returned 0
```

If no files were deleted, the system displays the following:

```
There are no deleted files in your system
```

Note:

In order to run the **purge_volumes** utility at boot time on Linux platforms, add the following line to the appropriate startup script on your Linux system, or add it to the **rc** script created by the postinstallation routine for the database configuration you want this activity to work against:

```
purge_volumes -u=admin-user -p=admin-password -g=dba -f
```

This starts **purge_volumes** at boot time and disables confirmation of each delete action.

report_volume

Lists the operating system path of all existing Teamcenter volumes. The path does not include the network node name.

SYNTAX

```
report_volume [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=file-name [-date=yymmddhhmmss] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the name of the output report file. The utility automatically appends this file name with the **.volumes** extension.

-date

Specifies the date string. Must be in *yymmddhhmmss* format where *yy* is the year, *mm* is the month, *dd* is the day, *hh* is the hour, *mm* is minutes and *ss* is seconds.

This argument is optional. If not supplied, all volumes are reported. Otherwise, only volumes and files created after the input date are reported.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Do not include a directory path with the **-f** argument. The output file must be written to the current working directory.

EXAMPLES

To list all existing volumes in a file called **out.volume**, enter the following on a single line:

```
$TC_ROOT/bin/report_volume -u=admin-user -p=admin-password -g=dba -f=out
```


review_volumes

Allows you to view volume file attributes regarding OS volumes (file size, last modification date, and so on) and to remove unreferenced operating system files from these volumes. This utility can generate a report file describing volume usage by various groups and users, as well as reporting any unreferenced operating system files, missing operating system files, and unreferenced Teamcenter files. Unreferenced operating system files can be deleted at the time a report file is generated or at a later time using a previously-generated report file as an input. The report file format is plain text (ASCII) and can be manually edited in order to not delete certain files. To prevent files from being deleted, remove any file names before using the report file as input. You can also save any deleted files to a ZIP format compressed file.

Note:

Before creating volumes, you must have an FMS server cache (FSC) installed and running, and you must set the **FMS_BootStrap_Urls** preference with the FSC host and port information.

The report file contains three main sections that list issues with Teamcenter files:

- Section 1: Unreferenced OS files

Lists files in the OS file system over a day old that are not referenced by Teamcenter. These are files that exist in the OS volume directory but are not referenced by any Teamcenter **TcFile** data objects. These are the OS files that are deleted when the **review_volumes** utility is run with the **-if** argument. During the **-if** run, these files are deleted only after a double-check that they are indeed not referenced.

- Section 2: Missing OS files

Lists files missing from the OS file that are referenced by Teamcenter. These are files that do not exist in the OS volume directory but are referenced by Teamcenter data objects. This situation may be remedied by copying the missing files from backed-up volumes or whole file caches. The **-if** run does not take any additional action on files reported in this section.

- Section 3: Unreferenced Teamcenter files

Lists files in the OS file system referenced from unreferenced **ImanFile** objects. These are files that exist as **TcFile** data objects, but those **TcFile** data objects are not referenced by any dataset data objects. For files that are listed in this section, the **-if** run deletes the **TcFile** object and the OS file from the volume. Teamcenter referential integrity ensures that the **TcFile** object deletions occur only if they are not referenced by other Teamcenter objects.

To illustrate the report output, assume that the Teamcenter database has the following contents:

Dataset	Reference type	Referenced TcFile	Volume file name of the referenced file
MyDataset1	-	-	-
MyDataset2	Text	MyFile.txt	eng_3219879864/ MYFILE209734EA_32470987.txt
MyDataset3	MSWordX	Schedule3.docx	eng_3219879864/ SCH4867986324_3467.docx
	Zip	MaintenanceSchedules.zip	eng_3403297097f/ MAINTE49907r3732498_kjlewuih.zip
MyDataset4	File	4GMatrix.dat	eng_3403297097f/ 4GMAFJKSN39479679_3234777429869.dat
-	-	Lolapalooza.mid	eng_3219879864/ LOLAP214097409_32209874986.mid
-	-	VennDiagram.dia	eng_3219879864/ VENND329879864_379407.dia

Also assume that the volume contains the following files:

```
eng_3219879864/LOLAP214097409_32209874986.mid
eng_3219879864/MYFILE209734EA_32470987.txt
eng_3403297097f/MAINTE49907r3732498_kjlewuih.zip
eng_3403297097f/BARTFARGLE0321970_32986.tag
eng_3403297097f/FASTNR2139840397_3219473209.prt
```

The report lists the following:

- Unreferenced OS files

The following files exist in the volume but are not referenced by any Teamcenter data objects:

```
eng_3403297097f/BARTFARGLE0321970_32986.tag
eng_3403297097f/FASTNR2139840397_3219473209.prt
```

- Missing OS files

The following files do not exist in the volume but are referenced by Teamcenter data objects:

```
eng_3219879864/SCHED4867986324_3467.docx
eng_3403297097f/4GMAFJKSN39479679_3234777429869.dat
eng_3219879864/VENND329879864_379407.dia
```

- Unreferenced Teamcenter files

The following files exist as **TcFile** data objects, but these **TcFile** data objects are not referenced by any dataset data objects:

eng_3219879864/LOLAP214097409_32209874986.mid
 eng_3219879864/VENND329879864_379407.dia

SYNTAX

```
review_volumes [-u=user-id {-p=password | -pf=password-file} -g=group]
-v=volume -rf= file-name | -if=file-name
[-of=file-name] [-zf=file-name] [ -lv]
[-parallel=number-of-parallel-processes] [-rfolder=folder-name]
[-noDeleteCheck]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-v

Runs the utility against a single, specified volume. This argument is required unless the **-lv** or **-parallel** argument are specified.

-rf

Creates a report file listing unreferenced files in volumes.

Note:

The **-of**, **-if**, and **-zf** arguments do not work in combination with the **-rf** argument. When the **-rf** argument is present, only the report file generation is performed.

-if

Specifies the report file to be used as input to delete unreferenced files in volumes.

-of

Deletes and logs the results to a specified file. This argument must be supplied if the **-if** argument is used.

-zf

Saves deleted files to the specified **ZIP** file. The **.zip** extension is automatically appended to the file name if another extension is not specified.

-lv

Lists all volumes defined in the database.

-parallel

Specifies the number of volumes on which to simultaneously run this utility. You can use this argument to generate reports on all volumes defined in your database simultaneously (use the **-lv** argument to determine the number of volumes defined in your database). However, you must consider available computing resources while setting this value. Even if you have 800 volumes defined in your database, you might only have enough computing power to run five or ten processes in parallel.

This argument must be used with the **-rfolder** argument.

-rfolder

Specifies the folder in which the multiple reports generated by the **-parallel** argument are stored.

This argument must be used with the **-parallel** argument.

-noDeleteCheck

Turns off double-checking of OS files to be deleted. Double-checking ensures that OS files to be deleted are checked again for **ImanFile** references before deletion. This avoids accidental deletions when corrupt or tampered report files are used. Use this argument with the **-if** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in Manually configure the Teamcenter environment..

FILES

As specified in Log files produced by Teamcenter.

RESTRICTIONS

In versions prior to 8.3.2, you could supply the **-rf** argument in combination with the **-of** argument in order to have the **review_volumes** utility simultaneously generate a report file and delete files from the specified volume.

Beginning with Teamcenter 8.3.2, this requires two separate steps:

1. Run the **review_volumes** utility with the **-rf** argument.
2. Use the **-if** and **-of** arguments to input the previously created **report** file and **delete volume** files.

This change in how arguments are processed was required by the need to improve scalability and performance of the **review_volumes** utility, for example, the addition of the new **-parallel** argument.

EXAMPLES

- To generate a report on a single volume, enter the following command on a single line:

```
review_volumes -u=user-id -p=password -g=group -v=volume -rf=file-name
```

- To delete files on a single volume from a previously executed report, enter the following command on a single line:

```
review_volumes -u=user-id -p=password -g=group -v=volume -if=file-name -of=file-name
```

- To list all volumes defined in the database, enter the following command on a single line:

```
review_volumes -u=user-id -p=password -g=group -lv
```

- To generate report on all volumes defined in the database, enter the following command on a single line:

```
review_volumes -u=user-id -p=password -g=group -parallel=number-of-parallel-processes  
-rfolder=folder-name
```

syncCache

Loops through each dataset row for the specified applications in the cache database and performs the following tests and actions:

- If the **Dataset.Folder** field is either null or empty, the utility performs no action.
- If the **Dataset.Folder** field does not exist or is not a directory, the dataset is deleted from the cache.¹
- If the **Dataset.AppRef** field is not found in Teamcenter, the dataset is deleted from the cache.²
- The dataset and its related **ItemRev** and **Item** rows in the cache database are updated from Teamcenter. This process performs most of the **Open** process except for downloading the design file. Since no files are downloaded, the staging directory is not modified.

SYNTAX

syncCache *application-name* [-help]

ARGUMENTS

application-name

Specifies a list of ECAD application names so that only the cached datasets created by the applications in the list are processed.

-help

Displays help for this utility.

ENVIRONMENT

Requires the same environment as for running EDA.

FILES AND RESTRICTIONS

None.

EXAMPLES

- To run **syncCache** on datasets created by Cadence Allegro:

```
syncCache cadenceSchematic cadencePcb
```

1 When a dataset is deleted from the cache, then the dataset's related ItemRev and Item are also deleted unless they are referenced by other datasets or ItemRev's.
 2 When a dataset is deleted from the cache, then the dataset's related ItemRev and Item are also deleted unless they are referenced by other datasets or ItemRev's.

- To run **syncCache** on datasets created by Mentor BoardStation:

```
syncCache mentor
```

tcmemstat

Monitors the Teamcenter model event manager (TcMEM).

By default, this utility is stored in the `TC_ROOT/tccs/bin` directory.

SYNTAX

tcmemstat [-h] [-x] [-status] [-restart] [-start] [-stop] [-kill]

ARGUMENTS

none

Displays TcMEM component versions and run time, if running.

-help or -h or -?

Displays help for this utility.

-x

Displays a summary of TcMEM status.

-status

Displays a complete TcMEM status report

-restart

Stops (if running) and restarts TcMEM, effectively reloading the configuration.

-start

Starts TcMEM if it is not already started.

-stop

Shuts down the TcMEM process immediately if no other clients are connected or if all connected clients are idle. Otherwise, a warning message is displayed.

-kill

Immediately and unconditionally shuts down the TcMEM process.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

tspstat

Monitors the Teamcenter server proxy.

By default, this utility is stored in the `TC_ROOT/tccs/bin` directory.

SYNTAX

tspstatt [-h] [-x] [-status] [-config] [-reconfig] [-restart] [-start] [-stop] [-kill]

ARGUMENTS

none

Displays **TcServerProxy** component versions and run time, if running.

-help or -h or -?

Displays help for this utility.

-x

Displays a summary of **TcServerProxy** status.

-status

Displays a complete **TcServerProxy** status report.

-config

Displays the name of **TcServerProxy** configuration file.

-reconfig

Reloads the **TcServerProxy** configuration. Use this argument to update a running **TcServerProxy** with any HTTP/HTTPS proxy changes.

-restart

Stops (if running) and restarts **TcServerProxy**, effectively reloading the configuration.

-start

Starts **TcServerProxy** if it is not already started.

-stop

Shuts down the **TcServerProxy** process immediately if no other clients are connected or if all connected clients are idle. Otherwise, a warning message is displayed.

-kill

Immediately and unconditionally shuts down the **TcServerProxy** process.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

vms_upgrade

Modifies any commercial part, vendor part, and vendor data created before Teamcenter 2007.1 MP4 to conform to the updated vendor management data model.

The utility creates GRM relations between these items where a **VendorIdentifier** object had previously been used. The **CommercialPart** object is now directly related to the **ManufacturerPart (Vendor Part)** object by the new **VMRepresents** relation, which holds vendor status information. The **VendorIdentifier** and **IdContext** objects are no longer associated with the **CommercialPart** object. The new **TC_vendor_part_rel** GRM relation is used to associate the **ManufacturerPart** with the vendor.

SYNTAX

```
application_root/bin/vms_upgrade [-u=user-id {-p=password | -pf=password-file} -g=group] -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

unmigrate_from_hsm

Reverses migration of the existing Teamcenter volume files objects from the purview of hierarchical storage management (HSM) by removing the **hsm_info** object associated with file objects from the database. Use this utility for the following situations:

- Encountering errors while migrating files to HSM.
- Maintenance requirement to remove a specific volume from purview of HSM.
- Remove all volumes from the purview of HSM.

After the execution of this utility completes, all physical volume files must be brought back to a primary tier in the event it has already migrated by third-party HSM software to secondary or tertiary tier. Siemens Digital Industries Software recommends sites execute this utility on a specific volume when no other users are accessing Teamcenter, especially during maintenance or upgrades. This utility can take considerable time to reverse migrate all files from the purview of HSM.

SYNTAX

```
unmigrate_from_hsm [-u=user-id {-p=password | -pf=password-file} -g=group]
[-listvolumes] [-volume= ALL | volume-name] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-listvolumes

List all volumes defined for the database.

-volume

Specifies particular volume to unmigrate. The system administrator can specify **volume=ALL** to unmigrate all volumes from the purview of HSM.

-v

Verbose mode. Provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

This utility is intended only for system-level users.

RETURN VALUES

Return value upon success

0

Return value upon failure

1

EXAMPLES

- To unmigrate all volumes defined for the database:

```
unmigrate_from_hsm -u=admin-user -p=admin-password -g=dba -volume=ALL -v
```

- To unmigrate a specific volume:

```
unmigrate_from_hsm -u=admin-user -p=admin-password -g=dba  
-volume=volume_name
```

- To list volumes names defined for a database:

```
unmigrate_from_hsm -u=admin-user -p=admin-password -g=dba -listvolumes
```


upgrade_vendor_part

Populates the new typed reference attribute in the **ManufacturerPart** from the existing data during upgrade. The associated vendor for the new typed reference attribute is retrieved from the **TC_Vendor_part_rel** relation.

SYNTAX

```
upgrade_vendor_part [-u=admin-user -p=$TC_USER_PASSWD] -g=dba -h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the user and password arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-g

Specifies the group associated with the user, in this case, **dba**

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

vm_report

Evaluates any or all volume management policies to generate a report. Because of performance considerations using the rich client application interface, Siemens Digital Industries Software recommends the system administrator execute this utility to create pending migration file sets.

Sites can schedule this utility to execute overnight through a Linux **cron** job or Microsoft Windows **at** command. This utility can take considerable time to evaluate all migration policies.

SYNTAX

```
vm_report [-u=user-id {-p=password | -pf=password-file} -g=group]
[-listvolumes] [-listpolicies]
[-migrationreport=migration-policy] [-filepath=filepath]
[-migrationreport=ALL {-srcvol=source-volume-name
-destvol=destination-volume-name} -filepath=filepath]
[-before=before-date] [-after=after-date] [-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-listvolumes

Indicates that the utility is to list all volumes in the database.

-listpolicies

Indicates that the utility is to list all active policies.

-migrationreport

Specifies migration policy for which the report need to be reported.

To generate a report on all active migration policies, specify **-migrationreport=ALL**. If you specify **ALL** for the migration policy, you must include either the **-srcvol** or **-destvol** arguments.

-srcvol

Specifies source volume name.

-destvol

Specifies destination volume name.

-before

Specifies the beginning date for reporting migration. For example, **23_Mar_2004**.

-after

Specifies the ending date for reporting migration. For example, **23_May_2006**.

-filepath

Specifies the operating system file path to which the report is saved.

-v

Verbose mode. Provides information about results and progress.

-h

Displays help for this utility.

ENVIRONMENT

- The generic command window set with all Teamcenter-related environments.
- As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

Due to performance considerations, Siemens Digital Industries Software recommends a system administrator execute this utility instead of using the rich client application interface.

EXAMPLES

- Execute the following command to generate report on all policies in the database:

```
vm_report -u=admin-user -p=admin-password -g=dba -migrationreport=ALL  
-srcvol=volume-one -destvol=volume-two -filepath=c:\temp\report.txt
```

- Execute the following command to generate report about a specific policy in the database:

```
vm_report -u=admin-user -p=admin-password -g=dba  
-migrationreport=policy-name -filepath=c:\temp\report.txt
```

- Execute the following command to list of all active policies names defined for a database:

```
vm_report -u=admin-user -p=admin-password -g=dba -listpolicies
```

volume_info

Displays volume information, including the network nodes containing the volumes.

SYNTAX

```
volume_info -u=user-id {-p=password | -pf=password-file} -g=group
[-volume=volume-name]
[-list_node]
[-list_volumes] -node=node-name
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally a user with administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-volume=*volume-name*

Displays information for the specified volume.

-list_node

Displays the current volume node.

-list_volumes

Displays a list of all Teamcenter volumes in the Teamcenter environment.

-list_volumes -node=*node-name*

Displays the volume on the specified node.

-h

Displays help for this utility.

EXAMPLES

- Enter the following command on a single line to list all the volumes in the Teamcenter environment:

```
volume_info -u=admin-user -p=admin-password -g=dba -list_volumes
```

- Enter the following command to display the information for the specified volume:

```
volume_info -u=admin-user -p=admin-password -g=dba -volume=volume-name
```

- Enter the following command to list the node for the current volume:

```
volume_info -u=admin-user -p=admin-password -g=dba -list_node
```

- Enter the following command to list the volume for the specified node:

```
volume_info -u=admin-user -p=admin-password -g=dba -list_volumes -node=node-name
```

xml_validator

Checks the XML file against the document type definition (DTD) to which it should conform.

SYNTAX

```
xml_validator  
[-v=always | never | auto*] file.xml
```

ARGUMENTS

-v

Specifies the validation scheme: **always**, **never**, **auto***. If not explicitly stated, defaults to **auto***.

file.xml

Specifies the XML file to be validated.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

EXAMPLES

1. The following example checks only the structure of the XML file:

```
xml_validator -v=never file1.xml
```

2. This example produces an error if the XML file does not have an associated DTD file. It always checks the validity of the XML file.

```
xml_validator -v=always file1.xml
```

3. This example checks the structure of the XML file if it does not correspond to a DTD. If the file corresponds to a DTD, it checks the validity of the file's XML against that of the DTD.

```
xml_validator file1.xml
```

File Management System (FMS) utilities

fccstat

Monitors the local FMS cache (FCC). By default, this utility is stored in the **\$FMS_HOME/bin** directory.

If no arguments are given **fccstat** reports whether FCC is online, FCC versions, and run time (if running).

SYNTAX

fccstat

```
[ -x ]
[ -status ]
[ -config ]
[ -performance [cloud] ]
[ -reconfig ]
[ -purge ]
[ -clear ]
[ -restart ]
[ -start ]
[ -stop ]
[ -kill ]
[ -h ]
```

ARGUMENTS

-x

Displays an FCC cache statistics summary, including whole file read, whole file write, and segment cache statistics. In addition, this option displays all offline FSC connections. Offline FSC connections are those that have been attempted and failed but have not yet been restored to service.

-status

Displays an FCC status and statistics summary, including whole file read, whole file write, and segment cache statistics, as well as client request statistics, FSC upload and download statistics, and the currently active assigned FSC. Assigned FSCs are listed as active by default, even if they have never been used. FSC addresses that have been attempted and failed, but have not yet been restored to service, are reported as offline.

-config

Displays the name of the local FCC configuration file used for bootstrapping.

-performance

Displays statistics on network parameters such as latency and bandwidth between the current FSC and the client (FCC) it is connected to. These measurements provide an indication of the network conditions that can be used to diagnose issues with performance.

Be aware that these values can vary widely based on network load, time of day, number of hops between the FCC and FSC, and so on. Therefore, the values reported using **-performance** should

not be taken as an absolute measure of performance. Instead, use them only to troubleshoot performance issues.

Use the **cloud** argument with **-performance** to report performance statistics when communicating with cloud (DSS) based volumes.

The disk or cloud volume used to measure upload and download performance is arbitrary. You cannot specify a particular volume.

-purge

Purges all files from the FCC cache, including the segment cache extent files.

-clear

Purges the cache completely. This removes all data but retains the segment cache extent files.

-reconfig

Reloads the FCC configuration. Use this option to update a running FCC with changes made to the local FCC XML configuration files.

-restart

Stops (if running) and restarts the FCC, effectively reloading the configuration. Environment variables still override any configuration file settings or changes.

-start

Starts the FCC if it is not already started.

-stop

Shuts down the FCC process immediately if no other clients are connected or if all connected clients are idle. Otherwise, a warning message is displayed.

-kill

Immediately and unconditionally shuts down the FCC process.

-h

Displays help for this utility. Help can be localized if the FCC is running.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the following:

- The **FMS_HOME** variable must be set to a valid FMS server directory.
- The **JAVA_HOME** variable must be set to a valid Java SDK directory.
- FCC must be running.
- This command must be run in an FMS operating environment.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To determine whether the FCC is running, run the **fccstat** command. For example:

```
TC_ROOT\tccs>fccstat
```

Example output:

```
fccstat: FCC offline.
```

- To display help for this utility and a statistic summary, run the **fccstat** command with the **status** argument. For example:

```
TC_ROOT\tccs>>fccstat -? -x -status fccstat -?:
```

Example output:

```
FMS Client Cache Status Utility
fccstat command line options:
  no options      Display FCC component versions and execution time.
  -help           Display this help message.
  -h              Display this help message.
  -?              Display this help message.
  -x              Display FCC status summary.
  -status         Display complete FCC status.
  -config         Display FCC configuration filename.
  -purge          Clear the FCC Cache and remove extents.
  -clear          Clear the FCC Cache and retain extents.
  -remove (UID)   Remove a UID from the FCC Cache.
  -restart        Stop and restart the FCC.
  -stop           Stop the FCC if no clients are connected.
  -kill           Stop the FCC even if clients are connected.

fccstat -x:
Cache:
  segment: 1 files, 507904 bytes.
  read:    38 files, 268433410 bytes.
  write:   0 files, 0 bytes.
Servers:
  0 files downloaded.
```

```

Active assigned FSC is 'http://127.0.0.1:4444/'
fccstat -status:
Cache:
  segment: 1 files, 507904 bytes, 0 hits, 0 misses.
  read:    38 files, 268433410 bytes, 0 hits, 0 misses.
  write:   0 files, 0 bytes.
Clients:
  3 Client connections established.
  6 Client request messages processed.
  5 Client response messages processed.
  0 Client status messages processed.
  0 Client error messages processed.
Servers:
  0 segments downloaded.
  0 files downloaded.
  0 files uploaded.
Active assigned FSC is 'http://127.0.0.1:4444/'

```

- To display the name of the local FCC configuration file and clear the cache, run the **fccstat** command with the **clear** argument. For example:

```
TC_ROOT\tccs>>fccstat -config -clear
```

Example output:

```

fccstat -config:
Z:\pkmvob1\fms\build\install\fms\fcc.xml
fccstat -clear:
Cache cleared.

```

- To purge a UID from the cache, run the **fccstat** command with the **remove** argument. For example:

```

TC_ROOT\tccs>fccstat
-remove abcd1234fedc9876ef005678ba005432

```

Example output:

```

fccstat -remove:
UID abcd1234fedc9876ef005678ba005432 has been removed from

the FCC Cache.

```

- To stop the FCC, run the **fccstat** command with the **stop** argument. For example:

```
TC_ROOT\tccs>>fccstat -stop
```

Example output:

```
fccstat -stop:
FCC Stopped.
```

- To restart the FCC, run the **fccstat** command with the **restart** argument. For example:

```
TC_ROOT\tccs>>fccstat -restart
```

Example output:

```
fccstat -restart:
FCC Started.
```

- To shut down the FCC process immediately when no clients are connected or all clients are idle, run the **fccstat** command with the **stop** argument. For example:

```
TC_ROOT\tccs>>fccstat -stop
```

Example output:

```
fccstat -stop:
FCC Stopped.
```

- To kill the FCC, run the **fccstat** command with the **kill** argument. For example:

```
TC_ROOT\tccs>>fccstat -kill
```

Example output:

```
fccstat -kill:
FCC Stopped.
```

- To print an FCC status and statistics summary, including whole file read, whole file write, and segment cache statistics, as well as client request statistics, FSC upload and download statistics, and the currently assigned FSC, run the **fccstat** command with the **status** argument. For example:

```
TC_ROOT\tccs>fccstat -status
```

Example output:

```
fccstat -status:
Cache:
  segment: 19 files, 52854784 bytes, 590722 hits, 1781 misses.
  read:    19 files, 134216705 bytes, 106253 hits, 38 misses.
  write:   0 files, 0 bytes.
Clients:
  5 Client connections established.
  388741 Client request messages processed.
```

```
392284 Client response messages processed.  
79 Client status messages processed.  
1 Client error messages processed.  
Servers:  
839 segments downloaded.  
19 files downloaded.  
0 files uploaded.  
Active assigned FSC is 'http://127.0.0.1:4444/'
```

- Entering **fccstat -reconfig** and the operation is successful returns the following message:

```
FCC successfully reconfigured.
```

- Entering **fccstat -reconfig** and the operation is not successful returns the following message:

```
FCC reconfiguration failed:  
Cannot reconfigure FCC pipe name.
```

fms_file_uids

Finds the FMS file unique ID (UID) of the volume file associated with a dataset or absolute volume path. You can use this utility to fix corrupted files in the FMS cache or volume. Use the **-f=list** argument to obtain a list of file UIDs, and use the **-f=download** argument to download files from the FMS cache or a volume.

File corruption is caused generally due to network issues in the environment. If a file is found corrupt in the FSC cache, the FMS file UID can be used to purge the file from the FSC cache using the **fscadmin** utility. If the file is found corrupt in the volume but not corrupt in the FSC cache, the FMS file UID can be used to download the uncorrupted file from the FSC cache.

Once the file is recovered, it should be verified for correctness. This can be done by opening the file with the appropriate tool (for example, Microsoft Word or NX). Once it is verified as correct, it may simply be copied over the corrupted file in the volume. This may require permission changes depending upon what account has ownership.

You can recover files in the following ways:

- Recover a file from the FSC cache

The **fscadmin** utility can be used to verify whether a file UID has been previously cached, for example, **fscadmin -s http://address:port./cachedetail**. The console output looks similar to the following:

```
Cache details: FSC_tri6w035_nox-FSCReadMap
Files: 8, Bytes: 3260416, Hits: 0, Misses: 0
FSC_tri6w035_nox-FSCReadMap cache contains fragments of files with the following guids:
00000000000004a55091052da5548113, len: 1529282
0000000000000367508fadc3a5548113, len: 3292
0000000000000397508fadc3a5548113, len: 3904
000000000000037f508fadc3a5548113, len: 3759
00000000000003df508fadc3a5548113, len: 2269
0000000000000425509a5190a5548113, len: 1529282
0000000000000429509a6102a5548113, len: 96626
00000000000003af508fadc3a5548113, len: 1909
Cache details: FSC_tri6w035_nox-FSCWriteMap
Files: 2, Bytes: 1638400, Hits: 0, Misses: 0
FSC_tri6w035_nox-FSCWriteMap cache contains fragments of files with the following g
guids:
000000000000042c50913584a5548113, len: 89120
00000000000004845097e951a5548113, len: 1529282
Cache details: whole file disabled.
cachedetail done.
```

- Recover a file from the FCC cache

It may be possible to recover a file from the FCC cache. Once you run the **fms_file_uids** utility with the **-list** argument, you can determine the file UIDs for each file and search the FCC cache for that file by that UID.

For example steps (given a file UID of **00000000000021c84d2ba8e281f08a1f**):

```
cd %FCC_CACHE_DIR% dir /s fmsr_000000000000021c84d2ba8e281f08a1f.*
```

If this file is found in the FCC cache, it may be copied into the appropriate volume location in the same manner as you would copy a file recovered from FSC cache from the **localfolder** generated by the **fms_file_uids** utility.

SYNTAX

```
fms_file_uids [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=[list | download]
-v
-h
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file. If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

list Lists the FMS file UIDs with the volume file name, associated dataset name, and dataset type (if any).

This information is written to a file named **fmscache_gile_guids.txt**. Following is sample content:

```
FSC GUID = 000000000000217a50abb00620cc4bca, VolumeFile Name =
tst031_text_ev309kc4nflj8.txt
    Dataset Name = tst031, Dataset Type = Text
    Dataset Name = tst031, Dataset Type = Text
FSC GUID = 000000000000218e50abb00620cc4bca, VolumeFile Name =
tst032_text_i6309kc4nfltd.txt
    Dataset Name = tst032, Dataset Type = Text
    Dataset Name = tst032, Dataset Type = Text
FSC GUID = 000000000000224450abb00620cc4bca, VolumeFile Name =
tst033_text_66x09kc4nflxf.txt
    Dataset Name = tst033, Dataset Type = Text
    Dataset Name = tst033, Dataset Type = Text
FSC GUID = 000000000000222250c1d05c20cc4bca, VolumeFile Name =
tst034__wor_5jw059g4oap0j.doc
    Dataset Name = tst034_item/A, Dataset Type = MSWord
    Item Id = tst034_item, Item Revision Id = A
    Dataset Name = tst034_item/A, Dataset Type = MSWord
```

In this example, the first three files were created as named references to a dataset. The fourth file was created as part of a dataset under an item and item revision.

download Downloads files from the FSC cache into a subfolder named **localfolder**. The FMS file UIDs not found in cache are downloaded from the volume if an FSC has been configured to serve a volume.

The following subfunctions can be used with the **download** argument:

-fscURI=URI Specifies the Uniform Resource Identifier (URI) of the FSC cache where the UID is stored, that is, **http://address:port**. This is only used with the **-f=download** argument.

-volumePath List=filename Specifies the input file consisting of fully qualified volume filenames. This argument can be used instead of the **-datasetNameList** argument.

Note:

The **-volumePathList** argument contents can be generated by running the **move_volume_files** utility using the **-f=list** and **-srcvol=volume-name** arguments (among others). This creates a file with fully qualified file names that can be used as input to this utility.

-datasetNameList=filename Specifies the input file consisting of dataset names. This argument can be used instead of the **-volumePathList** argument.

-v

Specifies verbose mode.

-help or -h or -?

Displays help for this utility. Help can be localized if the FCC is running.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment* and the following:

- The **FMS_HOME** variable must be set to a valid FMS server directory.
- The **JAVA_HOME** variable must be set to a valid Java SDK directory.
- FCC must be running.
- This command must be run in an FMS operating environment.

Configure two FSCs, one as a cache server and one as a volume server. For example:

```
<fmworld>
  <fmsenterprise id="123456" volumestate="normal">
    <fscdefaults>
      ...
    </fscdefaults>
    <fccdefaults>
      ...
    </fccdefaults>
    <fscgroup id="mygroup1">
      <fsc id="myfsc1" address="http://myhost;6789" ismaster="true">
        <volume id="123abc" enterpriseid="123456" root="d:\myvolume" priority="0" />
      </fsc>
    </fscgroup>
    <fscgroup id="mygroup2">
      <fsc id="myfsc2" address="http://myhost:1234" ismaster="true"/>
      <clientmap subnet="127.0.0.1" mask="0.0.0.0">
        <assignedfsc fscid="myfsc2" priority="0" />
      </clientmap>
    </fscgroup>
  </fmsenterprise>
</fmworld>
```

```

    </fscgroup>
  </fmsenterprise>
</fmsworld>

```

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- A functioning FSC is required to use this utility.
- To use the utility with older saved FSC caches, an FSC should be configured solely for the purpose of recovering files from the cache.
- Sometimes the file may have been partially cached in segment cache. In such an event, it attempts to pull down the missing segments from the volume file (if available). If the file has not been previously cached, the utility exports the file from the volume (again, if configured and available).
- When running the recovery utility, no volume should be configured in FMS. If a volume is configured there is no way of knowing if a recovered file came from cache or from the volume. Without a volume configured, the utility reports an error if the file cannot be found in either location.

EXAMPLES

- The following example writes the list of UIDs to a file:

```
-fms_file_uids -u=username -p=password -g=dba -f=list -volumePathList=c:\myfilelist.txt
```

- The following example downloads files from the FSC cache:

```
-fms_file_uids -u=username -p=password -g=dba -f=download -fscURI=http://myhost:1234
-datasetNameList=C:\mydatasetlist.txt
```

fscadmin.sh/.bat

Monitors and controls File Management System FSC servers. This can be used to check the status of a server, perform a shutdown, modify logging levels, query performance counters, or to clear or inspect caches.

SYNTAX

\$FMS_HOME/fscadmin.sh
[-h] [-k *keyfile*] [-s *serveraddr*] [-f *tickets-file*] [command]

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-h

Displays help for this utility.

-k

Specifies a file containing the encryption key required by this system. A key file is a text file containing an ASCII-HEX encryption key. This is generally the same key file referenced in the **fmsmaster.xml** file for this system.

This argument is optional.

Example: **fscadmin -k site123keyfile.txt ./status**
 Default: **<none>**

-s

Specifies the protocol server for the FSC and the port you wish to communicate with.

Example: **fscadmin -s http://myserver:4445 ./status**
 Default: **http://127.0.0.1:4444**

-f

Specifies the name of the tickets file.

command

Command is a formatted string with the following fields:

FSCID/FUNCTION[/SUBFUNCTION/...]

FSCID

The FSC with the given ID, as defined in the primary configuration, for which this command is intended. A period (.) can be used to indicate the local (current) FSC you are connecting to, as indicated by the **-s** parameter.

FUNCTION [/SUBFUNCTION/...]

The functions and subfunctions are enumerated in [Function\[/Subfunction/...\] Details](#), later in this section.

Example: **fscadmin -s http://myserver:4445 ./status**

Example: **fscadmin -s http://myserver:4445 ./log**

Default: **./status**.

Example: **fscadmin -s http://myserver:4444 fsc123/log**

ENVIRONMENT

- The **FSC_HOME** variable must be set to a valid FMS server directory.
- The **JAVA_HOME** variable must be set to a valid Java SDK directory.

FILES

- **fscadmin.properties**

(Optional) A property file used to configure proxy and SSL options if required by the **fscadmin** utility.

- **fscadmin.properties.template**

A template of the available properties that can be set for the **fscadmin** utility.

RESTRICTIONS

- You can route **fscadmin** commands to remote FSCs without having to specify that FSCs address on the command line.
- Input and output through the **fscadmin** utility is not localized. Commands are subject to change without notice.

FUNCTION[/SUBFUNCTION/...] DETAILS

FUNCTION[/SUBFUNCTION]	Description
cachesummary	Summary of the read and write caches (number of files, bytes, hits, misses).
cachesummary/read	Summary of the read cache.
cachesummary/write	Summary of the write cache.
cachesummary/whole	Summary of the whole file cache.
cachedetail	Summary and detail of the read and write caches (GUIDS, filesizes).
cachedetail/read	Summary and detail of the read cache.
cachedetail/write	Summary and detail of the write cache.
cachedetail/whole	Summary and detail of the whole file cache.
clearcache	Clears (empties) both read and write caches.
clearcache/read	Clears the read cache.
clearcache/write	Clears the write cache.
clearcache/whole	Clears the whole file cache.
config	Dumps the configuration for this FSC.
config/hash	Displays the MD5 hash for the current primary configuration file.
config/reload	Reloads the latest configuration from the usual configuration sources, either from disk or via download from another FSC.
config/reload/all	Performs a coordinated configuration reload first reloading all FSC configuration primaries, then reloading all FSC configuration secondaries. Returns final configuration hash results in a comma-separated list.
config/reload/slaves	Performs a coordinated configuration reload of all FSC configuration secondaries. Returns final configuration hash results in a comma-separated list.
config/report	Queries the configuration hashes from all FSCs and returns results in a comma-separated list.
filestoresummary	Summary of all filestores (volumeid, root path, files, dirs, bytes).

FUNCTION[/SUBFUNCTION]	Description
filestoresummary/xxx	Summary of a filestore specified by xxx.
filestoredetail	Summary and detail of all filestores (file name, length, last modified, last accessed).
filestoredetail/xxx	Summary and detail of a filestore specified by xxx.
loadfscache/status	Displays the simple Load FSC Cache status. This includes the number of tickets received, valid, invalid, expired, etc. It also includes the number of files that have been downloaded, failed to download, and are waiting to be downloaded.
loadfscache/status/reset	Resets the Load FSC Cache statistics.
loadfscache/queue	Displays the tickets remaining in the Load FSC Cache queue.
loadfscache/queue/clear	Removes all remaining tickets in the Load FSC Cache queue.
loadfscache/filelist	Uses -f option to upload a file of tickets that should be populated into the specified FSC.
loadfscache/stat	Returns the number of transactions processed by the Load FSC Cache context.
loadfscache/stat/valid	Returns the number of valid tickets processed.
loadfscache/stat/invalid	Returns the number of invalid tickets processed.
loadfscache/stat/expired	Returns the number of expired tickets processed.
loadfscache/stat/total	Returns the total number of tickets processed.
loadfscache/stat/ok	Returns the number of files successfully populated.
loadfscache/stat/failed	Returns the number of files which failed to populate.
loadfscache/stat/waiting	Returns the number of tickets remaining to be populated
loadfscache/stat/processing	Returns 1 if population is occurring, otherwise 0.
loadfscache/stat/localfiles	Returns the number of local files on which population was attempted, but was not required.
loglevel/show	Shows the current logging level of all loggers.
loglevel/[fatal error warn info debug]	Sets the log level for all loggers starting with com to xxx.

FUNCTION[/SUBFUNCTION]	Description
loglevel / <i>logger</i> /[fatal error warn info debug]	Sets error log level on the logger and all children. <i>logger</i> is in the form of the name of the class.package of which you wish to change the log level. com.teamcenter.fms.servercache is all of the server com.teamcenter.fms.servercache . FileHandleCacheManager sets only the loglevel on that class.
log	Dumps the current logfile contents.
migratecache	Migrates the whole files from segment cache to whole file cache.
migratecache/status	Prints the status of migratecache command.
purgecache	Purges the caches, reclaiming disk space.
purgecache/read	Purges the read cache.
purgecache/write	Purges the write cache.
purgecache/whole	Purges the whole file cache.
purgeguid / <i>xxx</i>	Removes the GUID (file) specified by <i>xxx</i> from the cache.
perfcounters	Reports the value of system performance counters.
perfcounters/reset	Resets the performance counters.
shutdown	Stops the FSC when it becomes idle (when all current file transfers are complete). The FSC will reject all new incoming file requests.
shutdown/maxwait / <i>xxx</i>	Stops the FSC when it becomes idle (when all current file transfers are complete) or when a maximum number of seconds have been exceeded. The FSC will reject all new incoming file requests. If the server does not become idle, <i>xxx</i> is the number of seconds to wait before forcing the shutdown.
shutdown/now	Stops the FSC.
status	Displays simple status about the FSC (FSCID, site, running time) and also prints the number of concurrent admin and file-based connections. This also shows the remaining time before a forced shutdown, if one is pending.
stop	Stops the FSC.

FUNCTION[/SUBFUNCTION]	Description
useragents	Prints a tally of all the useragents (clients) that have connected.
version	Prints the versions of the FSC JAR files.

EXAMPLES

- To see general statistics of the server, enter the following command:

```
./fscadmin.sh -s http://myserver:4445 ./status
```

Example output:

```
FSC id: myfsc, site:fms.teamcenter.com
running for 3 days, 17 hours, 3 min, 46 sec
Current number of file connections: 0
Current number of admin connections: 1
```

- To see general statistics of the caches, enter the following command:

```
./fscadmin.sh -s http://myserver:4445 ./cachesummary
```

Example output:

```
Cache summary: myfsc-FSCReadMap
Files: 0, Bytes: 0, Hits: 0, Misses: 0
Cache summary: myfsc-FSCWriteMap
Files: 0, Bytes: 0, Hits: 0, Misses: 0
```

- To see version information for the server, enter the following command:

```
./fscadmin.sh -s http://myserver:4445 ./version
```

Example output:

```
FMSServerCache version: 1.1, build date: 20050729
FMSUtil version: 1.1, build date: 20050729
FSCJavaClientProxy version: 1.1, build date: 20050729
```

- To set the FSC loglevel to **WARN**, enter the following command on a single line:

```
./fscadmin.sh -s http://myserver:4445 ./loglevel/warn
```

Example output:

loglevel done.

- To cause an idle shutdown, waiting no more than 1 hour, enter the following command on a single line:

```
./fscadmin.sh -s http://myserver:4445 ./shutdown/maxwait/3600
```

Example output:

```
FSC will shutdown when idle, or in 3600 seconds...
```

- The following command prepopulates FSC using the **loadfscache/filelist** command:

```
fscadmin -s http://server:port -f ticketsfile ./loadfscache/filelist
```

FSCWholeFileCacheUtil

The primary purpose of this utility is to purge files from the whole file cache. This utility purges the cache based on file age, subdirectory size, and/or available disk space. This utility also purges misplaced files (files that are stored in the wrong cache partition or subdirectory). This utility can also be used to clear entire cache partitions, or to redistribute cached files among a new list of partitions.

By default, this utility is stored in the *FSC_HOME/bin* directory.

SYNTAX

```
FSCWholeFileCacheUtil {-clear | -purge | -redistrib}
[-d=cache-root-directory-path] [-l=file-name]
[-maxf=maximum-files-per-subdirectory] [-maxage=maximum-file-age]
[-pctfree=percent-disk-free] [-temp=temporary-cache-directory-path]
[-to=destination-partitions] [-from=source-partitions] [-copy]
[-h]
```

ARGUMENTS

-clear

Clears all permanent cache files from the cache.

-purge

Performs a single purge cycle on the cache.

-redistrib

Redistributes files to new partitions. This is useful when partitions are added to (or removed from) the cache, when the cache is cloned (copied), or when the cache is moved to another location.

-d

Specifies the cache partition data path. Because each partition is purged or cleared independently, only one partition path is accepted here. If not set, the current directory is used.

Use with either the **-clear** or **-purge** arguments.

-l

Specifies the full path name of the log file. If not set, no logging is performed.

Use with either the **-clear** or **-purge** arguments.

-maxf

Specifies the maximum number of files per subdirectory. If the number of files stored in the subdirectory exceeds the configured amount, files are automatically purged to the maximum number of files specified.

Valid values are **1024** through **10240**. The default setting is **10240**.

Use only with the **-purge** argument.

-maxage

Specifies (in days) the maximum file age. All cache files older than the specified age are purged.

Valid values are **0** through **3650**. The default setting is **3650**. Setting this argument to **0** clears the entire cache.

Use only with the **-purge** argument.

-pctfree

Specifies the minimum percentage of disk space to remain free.

Valid values are **0** through **100**. The default setting is **0**. Setting this argument to **0** purges the cache by only the subdirectory count specified by the **-maxf** argument and the file age specified by the **-maxage** argument. Setting this argument to **100** clears the entire cache.

Use only with the **-purge** argument.

-temp

Specifies the directory location to store temporary cache purge data. The default setting is the directory set by the **-d** argument.

Use only with the **-purge** argument.

-to

Specifies the source cache partition data path list. Redistribution almost always involves multiple partitions. This is a comma-separated list of all of the cache partition paths to be included in the final FSC whole file cache at the conclusion of the redistribution, in the same format as the **FSC_WholeFileCacheLocation** property in the **<fscdefaults>** XML configuration. Some or all partition paths may appear in both the **-to** and **-from** arguments, except when **-copy** is also specified.

This argument is required when the **-redistrib** argument is specified; it is not optional, and has no default value.

Use only with the **-redistrib** argument.

-from

Specifies the source cache partition data path list. Because redistribution almost always involves multiple partitions, this is a comma-separated list of all of the cache partition paths that constitute the existing FSC whole file cache. The default setting is the partition list set by the **-to** argument.

Use only with the **-redistrib** argument.

-copy

Specifies that files are to be copied rather than moved. This argument is useful for cloning cache partitions as a complete set. Do not use if the **-from** and **-to** arguments have partitions in common.

Use only with the **-redistrib** argument, when the **-from** and **-to** arguments have no partitions in common.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To clear files from the partition, enter the following command:

```
FSCWholeFileCacheUtil -clear -d="J:\FSC_WFC\FSC_WFC_p2\0"
-l="C:\Program Files\Teamcenter\FSC\ClearCache.log"
```

This removes all files from the FSC whole file cache partition at **J:\FSC_WFC\FSC_WFC_p2\0** and logs the activity in the **C:\Program Files\Teamcenter\FSC\ClearCache.log** file.

The existing partition owner must run this utility to obtain permission to delete each file.

- To purge files from the partition, enter the following command:

```
FSCWholeFileCacheUtil -purge -d="//FSC_WFC/FSC_WFC_p1/3"
-maxf=5120 -maxage=60 -pctfree=35 -temp="/scratch/FSCWholeFileCachePurge.tmp"
```

This purges excess files from the FSC whole file cache partition at **//FSC_WFC/FSC_WFC_p1/3** such that:

- All files more than 60 days old are deleted (as set by the **-maxage** argument).
- No more than 5120 files remain in each subdirectory after the first pass (as set by the **-maxf** argument).
- At least 35% of the partition size are free after the second pass (as set by the **-pctfree** argument).

- Temporary purge files are stored in the **/scratch/FSCWholeFileCachePurge.tmp** directory (as set by the **-temp** argument).

The existing partition owner must run this utility to obtain permission to delete each excess file.

- To add a new partition, enter the following command:

```
FSCWholeFileCacheUtil -redistrib -from="J:\FSC_WFC\FSC_WFC_p1,K:\FSC_WFC\FSC_WFC_p2"
-to="J:\FSC_WFC\FSC_WFC_p1,K:\FSC_WFC\FSC_WFC_p2,L:\FSC_WFC\FSC_WFC_p3"
```

This adds a third partition **L:\FSC_WFC\FSC_WFC_p3** to an existing FSC whole file cache already stored on partitions at **J:\FSC_WFC\FSC_WFC_p1** and **K:\FSC_WFC\FSC_WFC_p2**. Some files may be moved from the existing partitions to the new partition as the data is redistributed among the three new partitions.

The three partitions must be approximately the same size (within 10%).

The existing partition owner must run this utility to redistribute each file properly and maintain file permission integrity.

- To split a cache from existing partitions, enter the following command:

```
FSCWholeFileCacheUtil -redistrib -from="//FSC_WFC/FSC_WFC_p1, //FSC_WFC/FSC_WFC_p2"
-to="//FSC_WFCCopy/p1, //FSC_WFCCopy/p2, //FSC_WFCCopy/p3" -copy
```

This copies and splits an FSC cache from the two existing partitions **//FSC_WFC/FSC_WFC_p1** and **//FSC_WFC/FSC_WFC_p2** to three new partitions **//FSC_WFCCopy/p1**, **//FSC_WFCCopy/p2**, and **//FSC_WFCCopy/p3**. The existing partitions at **//FSC_WFC/FSC_WFC_p1** and **//FSC_WFC/FSC_WFC_p2** remain and can continue to be used.

The two existing partitions must be approximately the same size as each other (within 10%).

The three new partitions must also be approximately the same size as each other (within 10%). They do not need to be the same size as the two original partitions.

The existing partition owner must run this utility to copy each file properly and maintain file permission integrity. The same user owns the destination files at the conclusion of the redistribution.

generate_file_digests

Generates the digests of a Teamcenter volume. You create a report file of digest information and use the report file to generate the digests for the Teamcenter volume.

The checksum digest is generated from the file contents and is used to validate the contents of the file have not be altered during transfer or while on disk. Teamcenter supports three algorithms to generate file digests **MD5**, **SHA-1** and **SHA-256**. Each have an increasing degree of protection against falsification and length of digest.

SYNTAX

```
generate_file_digests [-u=user-id {-p=password | -pf=password-file} -g=group]
-v=volume-name [-rf=file-name] [-if=file-name]
[-digestAlgorithm=algorithm] [-overwrite] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-v

Specifies the Teamcenter volume.

-rf

Specifies the report file that contains the digest information for all files in the volume. This file is used as input to the **-if** argument. This argument is not valid with the **-if** argument.

-if

Specifies the file that contains the digests to be generated. This argument is not valid with the **-rf** argument.

-digestAlgorithm

Specifies the digest algorithm to use when generating digests for files. Use with the **-if** argument. The valid values are **MD5**, **SHA-1** and **SHA-256**.

-overwrite

Specify this argument to create new digests for all the files in the file specified using the **-if** argument.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To save a report of digest information for all files in an existing volume in the **report_file.txt** file, enter the following on a single line:

```
generate_file_digests -u=admin-user -p=admin-password -g=dba
-rf=report_file.txt
```

To generate digests for all files in the **report_file.txt** file with the **SHA-1** algorithm, enter the following on a single line:

```
generate_file_digests -u=admin-user -p=admin-password -g=dba
-if=report_file.txt -digestAlgorithm=SHA-1
```


install_datasharekeys

Generates private keys for the Data Share Manager.

The Data Share Manager is used to monitor asynchronous file uploads and downloads, and the keys are used to authenticate the file uploads and downloads with the Teamcenter database.

This utility is initially called by the Teamcenter installation process to install the predefined private keys. After the initial installation, you can use this utility to perform functions such as exporting a public key to be used when installing the Data Share Manager.

For general information about private and public keys, see http://en.wikipedia.org/wiki/Public-key_cryptography.

SYNTAX

```
install_datasharekeys -u=user-id {-p=password | -pf=password-file} -g=group -f={activate | delete |
exp_pubkey | exp_pvtkey | gen_pvtkey | install | list | modify | seed} -file=input-file -reserve[-v] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

activate	Sets an existing reserve key as the active private key and sets the previous active private key as the reserve key. There is only one active private key in the system that is used by Teamcenter. The general process to move to a new active private key is as follows: 1. Create a new reserve private key using the -reserve argument with the install or seed argument. 2. Export public keys for the reserve key and install the key on client machines. 3. Switch to the new reserve private key using the -f=activate argument. 4. Delete the old private key (now marked as reserve) at any later time.
delete	Deletes the Data Share Manager private key from the database.
exp_pubkey	Exports the Data Share Manager public key. This generates *.x509 key file and a *.pem file. The administrator packages the *.x509 file into a keys.zip file and distributes it to end users to be used when installing the Data Share Manager. (The *.pem file is not needed in the ZIP file.) The *.x509 file must be included in the ZIP file in a keys directory so it can be properly read by installers.
exp_pvtkey	Exports the Data Share Manager private key.
gen_pvtkey	Generates a new private key that can be imported to the database using the modify argument.
install	Generates a new key and installs the Data Share Manager private key into the database.
list	Lists the name of the Data Share Manager private key in the database.
modify	Modifies the Data Share Manager private key in the database.

seed

Seeds or imports a new Data Share Manager private key into the database.

-file=input-file

Specifies a file name (with full path) that contains the private key to import using the **-seed** argument.

If this optional parameter is specified with the **-seed** argument, the utility uses the specified filename argument as the private key to be imported. If this parameter is not specified when using the **-seed** argument, the utility prompts the user to specify the filename for the private key to be imported.

-reserve

Indicates that the current function should work on the reserve key. This argument can be used with the following functions: **install**, **seed**, **modify**, **delete**, **exp_pubkey**, **exp_pvtkey**, or **gen_pvtkey**.

-v

Displays verbose output.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To export the public key from the database, enter the following command on a single line:

```
install_datasharekeys -u=username -p=password -g=dba -f=exp_pubkey
```

Provide this key in a ZIP file to users installing the Data Share Manager. The key authenticates the users with the Teamcenter database when using the Data Share Manager to upload or download files.

- To generate a new private key and use it to modify the private key in the database, enter the following command on a single line:

```
install_datasharekeys -u=username -p=password -g=dba -f=gen_pvtkey  
-f=modify
```

install_encryptionkeys

Creates encryption keys for the File Management System (FMS). This utility is initially invoked by the Teamcenter installation procedure to install the predefined encryption keys. After the initial installation, this utility can be used to list, modify, and delete encryption keys.

SYNTAX

```
install_encryptionkeys -u=Tc-admin-user {-p=password | -pf=password-file} -g=group
-f=install | list | modify | delete | install_mediator_key [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

install	Installs the default encryption keys in the database.
list	Lists the encryption keys currently installed in the database.
modify	Modifies one or more encryption keys in the database.
delete	Deletes the encryption keys from the database.
install_mediator_key	Prompts the user to enter the Teamcenter Security Services Mediator Password Value key. The key is installed in the EncryptionKey table.
-h	Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To install the default encryption keys, enter the following command on a single line:

```
install_encryptionkeys -u=Tc-admin-user -p=password -g=group -f=install
```

install_fms_keys

Creates and imports File Management System (FMS) public-private key pairs into the Teamcenter database. **install_fms_keys** also exports the FMS public key to a keystore file. This utility is initially invoked by the Teamcenter installation procedure to install the key pairs. After the initial installation, this utility can be used to modify, export, and delete key pairs.

SYNTAX

```
install_fms_keys -u=Tc-admin-user {-p=password | -pf=password-file} -g=group
```

```
-f={install [-overwrite] | import -file_name=file_name [-overwrite] | delete | exp_pubkey  
-file_password=key_pass}  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies one of the following functions:

install	Generates a new FMS key pair and installs it in the Teamcenter database.
	Specify -overwrite to replace an existing key pair in the database.
import	Imports a key pair from the PKCS12 file specified by -file_name and installs the key pair in the Teamcenter database.
	Specify -overwrite to replace an existing key pair in the database.
delete	Removes the existing FMS key pair from the Teamcenter database.
exp_pubkey	Exports the FMS public key as a *.p12 keystore file. Use -file_password to specify the password of the *.p12 key file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To generate a new private key in the Teamcenter database and replace the existing key, enter the following command:

To install the default encryption keys, enter the following command on a single line:

```
install_fms_keys -u=Tc-admin-user -p=password -g=group -f=install
-overwrite
```

- To export the public key corresponding to the private key in the Teamcenter database, enter the following command:


```
install_fms_keys -u=Tc-admin-user -p=password -g=group -f=exp_pubkey  
-file_password=key_pass
```

load_fccache

Prepopulates the FMS client cache (FCC) using ITK calls, resulting in improved cache performance. You can also use this utility to read the PLM XML file generated by the **bomwriter** utility, and then parse the UIDs and generate read tickets. The read tickets can be prepopulated on the target FCC process. Additional options include:

- Prepopulate the FCC based on dataset type.
- Manage integrated clash management (ICM) behavior.
- Copy JT files to a local staging area.
- Update the PLM XML file to point to local JT files.

SYNTAX

```
load_fccache [-u=user-id {-p=password | -pf=password-file} -g=group]
-f=list [-plmxml=file-name] [-dataset_type=dataset-type-name]
-f=load [-plmxml=file-name] [-input_file=file-name] [-dataset_type=dataset-type-name]
[-latest_version=yes | no] [-output_file=file-name] [-copy_out=directory]
[-output_plmxml=file-name] [-use_absolute_location=yes | no]
[-config=configuration-file] [-log_filename=log-file-name] [-log_types=levels]
[-copy_out_lifetime=value] [-lifetime_check=yes | no]
[-lifetime_check_interval=value] [-lifetime_process_limit=value] [-purge] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Runs either the **list** or **load** function.

The **list** function lists specified UIDs, which can then be used to prepopulate the FCC using the **load** function. The **list** function accepts the following input:

- **-plmxml=file-name**

Lists the UIDs of all datasets in the PLM XML file generated by the **bomwriter** utility.

- **-dataset_type=dataset-type-name**

Lists the UIDs of all datasets of the specified dataset type.

The **load** function prepopulates the FCC as specified by the following input:

- **-plmxml=file-name**

Prepopulates the FCC with all the datasets in the PLM XML file generated by the **bomwriter** utility.

- **-input_file=file-name**

Prepopulates the FCC based on the UIDs previously generated by the **list** function.

- **-dataset_type=dataset-type-name**

Prepopulates the FCC with all datasets of the specified dataset type.

-latest_version

Use **yes** to prepopulate the FCC with only the latest dataset version or **no** to use the version specified by the UID. The default value is **yes**.

Use this argument with the **load** function.

-output_file

Direct the output to a specific file by specifying a file name. If this argument is not used, the default output file is **stdout**.

-copy_out

Copy each downloaded file to a specific directory by specifying the directory name.

Use this argument with the **load** function.

-output_plmxml

If the **load** function is set to prepopulate the FCC with a PLM XML file, and the **-copy_out** argument is used to specify a specific directory location to which copies of each downloaded file is loaded, and then use this argument to update the PLM XML with the same directory location as specified by the **-copy_out** argument.

You must use this argument with the **load** function and the **-copy_out** argument.

-use_absolute_location

Use **yes** to create absolute location references in the output PLM XML file. For more information about configuring default settings, use **-h -config**.

-config

Specifies the file containing default settings. For more information about configuring default settings, use **-h -config**.

-log_filename

Specifies the file to which logging messages are printed. If not set, the default location is the **stderr** file. For more information about configuring logging messages, use **-h -log_types**.

-log_types

Specifies which types of messages are logged. Valid values are:

NONE

ERROR

WARNING

INFORMATION

DEBUG

PERFORMANCE

ALL

Combine log types using the plus (+) symbol as a delimiter. For example:

ERROR+WARNING+INFORMATION

For more information about configuring logging messages, use **-h -log_types**.

-copy_out_lifetime

Specifies the lifetime date of the files stored in the directory specified by the **-copy_out** directory. The directory is scanned for files that are older than the specified lifetime date. Use this argument to automatically clean up older files that are no longer used. This argument must be used in conjunction with the **-copy_out** argument.

Use **-h -config** to display more information about configuring default settings.

-lifetime_check

Use **yes** to scan the directory specified by the **-copy_out** directory during startup and perform a lifetime check. The lifetime check is performed in random order. This argument must be used in conjunction with the **-copy_out** and **-copy_out_lifetime** arguments.

Use **-h -config** to display more information about configuring default settings.

-lifetime_check_interval

Specifies how frequently to scan the directory specified by the **-copy_out** directory and perform a random cleanup based on the interval value. For example, if set to **10**, a lifetime check is performed once every ten executions.

This argument must be used in conjunction with the **-copy_out**, **-copy_out_lifetime** and **-lifetime_check** arguments. If the directory specified by the **-copy_out** argument contains many files, and if it is not important to check the lifetime on each execution, setting the **-lifetime_check_interval** argument can improve performance.

Use **-h -config** to display more information about configuring default settings.

-lifetime_process_limit

Specifies how long (in seconds) lifetime processing can continue. The lifetime check is performed in random order. If the directory specified by the **-copy_out** directory contains a large number of files, lifetime processing can be lengthy. You can use this argument to randomly process a subset of the files by setting a low value. This argument must be used in conjunction with the **-copy_out**, **-copy_out_lifetime** and **-lifetime_check** arguments.

Use **-h -config** to display more information about configuring default settings.

-purge

Purges all files from the directory specified by the **-copy_out** directory.

-h

Displays help for this utility. Use **-h -config** to display more information about configuring default settings. Use **-h -log_types** for more information about logging output.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

To ensure the JT files referenced in the **xyz001.plmxml** file are using the same dataset versions as when it was generated (rather than the latest version), enter the following command on a single line:

```
load_fcccach -u=username -p=password -f=load -plmxml=xyz001.plmxml  
-latest_version=no -copy_out=plmxml_cache  
-output_plmxml=xyz001_local.plmxml
```

load_fscache

Reads a PLM XML file, parses it for globally unique identifiers (GUIDs) and generates read tickets for files referenced in the PLM XML file. The read tickets can be stored in the operating system (OS) file. The utility can also be used to load the file server cache (FSC) of a target/distant FSC serving the same database using the read tickets file.

SYNTAX

```
load_fscache [-u=user-id {-p=password | -pf=password-file} -g=group]
{ [-f=list {-plmxml=file-name | -dataset_type=dataset-type-name} ] |
[-f=load -fsctargets=string { [-plmxml=file-name] | [-input_file=file-name] |
[-dataset_type=dataset-type-name} } } [-latest_version=yes | no]
[-output_file=file-name] [[-config=file-name][-log_filename=file-name]
[-log_types=log-level] [-h[-config] [-log_types] ]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Runs either the **list** or **load** function.

The **list** function creates a list of GUIDs for object in the specified PLM XML file, which can then be used to prepopulate the FSC using the **load** function. The **list** function requires one of the following input arguments:

- **-plmxml=***file-name*

Lists the GUIDs of all datasets in the specified PLM XML file.

- **-dataset_type=***dataset-type-name*

Lists the GUIDs of all datasets of the specified dataset type.

The **load** function prepopulates the FSCs specified in the **-fsctargets** argument using the information provided by one of the following input arguments:

- **-plmxml=***file-name*

Prepopulates the FSC with all the datasets in the specified PLM XML file.

- **-input_file=***file-name*

Prepopulates the FSC based on the GUIDs previously generated by the **list** function.

- **-dataset_type=***dataset-type-name*

Prepopulates the FSC with all datasets of the specified dataset type.

-latest_version

Specifies the dataset version to use to prepopulate the FSC. Set to **yes** to send the latest dataset version or **no** to use the version specified by the GUID. The default value is **yes**.

Use this argument with the **load** function.

-output_file

Directs the output to a specific file by specifying a file name. If this argument is not used, the default output file is **stdout**.

-config

Specifies the file containing default settings. For more information about configuring default settings, use the **-h -config** arguments.

-log_filename

Specifies the file to which logging messages are printed. If not set, the default location is the **stderr** file. To display information about configuring logging messages, use the **-h** and **-log_types** arguments together.

-log_types

Specifies which types of messages are logged. Valid values are:

NONE

ERROR

WARNING

INFORMATION

DEBUG

PERFORMANCE

ALL

Combine log types using the plus (+) symbol as a delimiter. For example:

ERROR+WARNING+INFORMATION

To display information about configuring logging messages, use the **-h** and **-log_types** arguments together.

-h

Displays help for this utility. Use the **-h** and **-config** arguments together to display more information about configuring default settings. Use **-h** and **-log_types** arguments together to display information about logging output.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

None.

EXAMPLES

- To display a list of GUIDs for datasets in the **abc001.xml** file:

```
load_fscache -u=username -p=password -g=dba -f=list
-plmxml=abc001.xml
```

- To prepopulate the FSC for **cv001** (identified with **fscid fsc_cv001_username**) with the datasets in the **abc001.xml** file:

```
load_fscache -u=username -p=password -g=dba -f=load
-plmxml=abc001.xml
-fsctargets=fsc_cv001_username
```

- To create a **readtickets.txt** file containing the GUIDs of the datasets in the **abc001.xml** file:

```
load_fscache -u=username -p=password -g=dba -f=list
-plmxml=abc001.xml
-output_file=readtickets.txt
```

- To prepopulate the FSC for **cv001** (identified with **fscid fsc_cv001_username**) with the datasets in the **readtickets.txt** file:

```
load_fscache -u=username -p=password -g=dba
-fsctargets=fsc_cv001_username
-f=load -input_file=readtickets.txt
```

- To create a **readtickets.txt** file containing the GUIDs of all the **CAEAnalysisDS** datasets:

```
load_fscache -u=username -p=password -g=dba -f=list
-dataset_type=CAEAnalysisDS -output_file=readtickets.txt
```

- To prepopulate the FSC for **cv001** (identified with **fscid fsc_cv001_username**) with the datasets in the **abc001.xml** file and store all types of log messages to the **load.out** file:

```
load_fscache -u=username -p=password -g=dba
-fsctargets=fsc_cv001_username
-f=load -plmxml=abc001.xml -log_types=ALL
-log_filename=c:\temp\load.out
```

update_fms_configuration

Updates the FMS configuration to reflect changes in volume definitions and other FMS configuration options. The import can be performed either through Teamcenter Environment Manager or the **admin data_import** command line utility.

SYNTAX

```
update_fms_configuration [-u=user-id {-p=password | -pf=password-file} -g=group]
-inputfile=path-to-input-XML-file
-volumes=volume-name
-fsc_id=fsc-ID
-filestore=filestore-ID
-load_balancer=load-balancer-ID
[-addTransientVolume]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-inputfile

Specifies the full path to the input text file containing the volume name and FSC ID separated by |.

Following is an example input file:

```
#delimiter=|
#volume_name|fsc_id|filestore|load_balancer
volume1|FSC_S0928933_jpip_1||
volume2|FSC_S0928933_jpip_1||
volume3|FSC_S0928933_jpip_1||
volume4||FSC_S0928933_jpip_2|
volume5||FSC_S0928933_jpip_2|
volume6||FSC_S0928933_jpip_2|
volume7|||FSC_S0928933_jpip_3
volume8|||FSC_S0928933_jpip_3
volume9|||FSC_S0928933_jpip_3
volume10|FSC_S0928933_jpip_4||
volume11|FSC_S0928933_jpip_4||
volume12|FSC_S0928933_jpip_4||
```

-volumes

Specifies a comma-separated list of volumes to be updated in the FMS primary configuration

-fsc_id

Specifies the FSC ID when the ID type is **FSC**.

-filestore

Specifies FSC ID when the ID type is **Filestore Group**.

-load_balancer

Specifies the FSC ID when the ID type is **Load Balancer**. When the new volume is created, you must specify where in the FMS primary configuration file the definition of this volume should be placed: the FSC element section, the filestore group section, or the load balancer section. This is determined by which of three parameters is used. Only one of the **-fsc_id**, **-filestore**, and **-load_balancer** arguments is mandatory.

-addTransientVolume

Adds the transient volume definition for the specified Teamcenter server process (Tcserver) context to the **fmsmaster** configuration. Run this utility on each Teamcenter server and pool server to add the transient volume specific to each server. Doing so eliminates the need to manually define and update the transient volumes using information obtained using the **backup_xmlinfo** utility.

-h

Displays help for this utility.

RESTRICTIONS

You must be a privileged user to run this utility. You must use either the **-inputfile** argument, or the **-volumes** argument and one of **-fsc_id**, **-filestore**, and **-load_balancer** arguments.

EXAMPLES

- To update the FMS primary configuration of FSC ID **FSC_S0928933_fsc1** with **volume1**, enter the following command on a single line:

```
update_fms_configuration -u=admin-user -p=admin-password -g=dba
-volumes=volume1 -fsc_id=FSC_S0928933_fsc1
```

- To update the FMS primary configuration of filestore group ID **FSG_S0928933_fsg1** with **volume1**, **volume2**, and **volume3**, enter the following command on a single line:

```
update_fms_configuration -u=admin-user -p=admin-password -g=dba
-volumes=volume1,volume2,volume3 -filestore=FSG_S0928933_fsg1
```

- To update multiple volumes with multiple FSC, filestore group, or load balancer IDs defined in a text file, enter the following command on a single line:

```
update_fms_configuration -u=admin-user -p=admin-password -g=dba -inputfile=d:\volumes.txt
```

- To add a new transient volume specific to the current pool server which has an FSC ID **FSC_S0928933_fsc1**, enter the following command:

```
update_fms_configuration -u=admin-user -p=admin-password -g=dba
-addTransientVolume -fsc_id=FSC_S0928933_fsc1
```


21. Schedule Manager Utilities

`schmgt_async_r11`

If your site has Schedule Manager data and you are upgrading from a Teamcenter version earlier than 11.2, you must manually run the Schedule Manager asynchronous attribute-consolidation utility after TEM completes the upgrade. If your site has no Schedule Manager data, you do not run this utility.

Until you run this utility, you cannot:

- Access Schedule Manager data. (However, other types of data are accessible before running the utility).
- Use TEM to add new features.
- Add data model customization to your environment.

This utility performs the following:

- Consolidates legacy attribute values from schedule revision, schedule task revision, execution form, scheduling form, and cost form into the attributes on the new **Workspace Object**-based schedule and schedule task.
- Consolidates resource assignments **ImanRelation** primary objects with schedule task revisions to the new **Workspace Object**-based schedule task and schedule calendar **ImanRelation** primary objects with schedule revisions to the new **Workspace Object**-based schedule.
- Consolidates any other **ImanRelation** primary objects with schedule task revisions to the new **Workspace Object**-based schedule task.
- Migrates data to the new schedule member's objects.

Syntax

Command to run the utility in data pre-check scan mode:

```
schmgt_async_r11 [-u=user-id {-p=password | -pf=password-file} -g=group]
                 [-m=dryrun]
                 [-h]
```

Command to run the utility in data consolidation mode:

```
schmgt_async_r11 [-u=user-id {-p=password | -pf=password-file} -g=group] ]
                 [-h]
```

Command to run the utility in data consolidation mode with input CSV file:

```
schmgt_async_r11 [-u=user-id {-p=password | -pf=password-file} -g=group]  
[-input_file=CSV file path containing Schedule UIDs]  
[-h]
```

Arguments

-u

Specifies the user ID. The user must have administrative privileges

-p

Specifies the user's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the user-ID value to be the password.

-pf

Specifies the password file.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the user-ID value to be the password.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

-m

Specifies the mode in which the utility will be executed in pre-check scan mode.

-input_file

Specifies the CSV file path containing Schedule UIDs.

-h

Displays help for this utility.



22. Systems Engineering utilities

add_req_templates

Installs the default Microsoft Word and Excel requirements management templates.

Note:

This utility usually runs during the installation and upgrade processes and does not need to be run manually. However, if the installation or upgrade does not install the templates, you can run this utility manually.

SYNTAX

```
add_req_templates [-u=user-id] [-p=password | -pf=password-file] [-g=group]
[-f=path-of-folder -t={SpecTemplate | ObjectTemplate | ExcelTemplate}]
[-i=path-of-file]
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-f

Specifies the path of the folder from which the templates are imported. Use this argument to import multiple templates from the same folder.

If you use this argument, you must also use the **-t** argument.

-t

Specifies the template type being imported. It is either **SpecTemplate**, **ObjectTemplate**, or **ExcelTemplate**.

If you use this argument, you must also use the **-f** argument.

-i

Specifies the path of the template file to be imported. Use this argument to import a single template.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

- To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
- The **-f** and **-t** arguments must be used together if they are used.

add_se_templates

Installs the system engineering templates, diagramming templates, and Visio templates required for Systems Engineering.

Note:

This utility usually runs during the installation and upgrade processes and does not need to be run manually. However, if the installation or upgrade does not install the templates, you can run this utility manually.

SYNTAX

```
add_se_templates [-u=user-id] [-p=password | -pf=password-file] [-g=group]
-dir=templates-directory
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-dir

Specifies the directory containing the Systems Engineering templates; for example, `{TC_INSTALL_DIR}/systemsengineering`.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.

req_convert_to_plaintext

Removes rich text, such as images, URLs, and OLEs, from requirements and converts the body text into plain text. It also applies the content type as plain text to the requirement revision.

SYNTAX

```
req_convert_to_plaintext [-u=user-id] [-p=password| -pf=password-file] [-g=group]  
-i=specification-element-id -r=revision-id  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-i

Specifies the specification element's item ID to convert.

-r

Specifies the revision's ID.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.

req_migrate_bomview_tracelinks

Starting in Teamcenter 10.1.2.1, trace links are created on revisions of the absolute occurrence objects. In earlier versions of Teamcenter, trace links are created on absolute occurrence objects.

If your database contains trace links on absolute occurrence objects, run this utility after upgrading the corporate server. This utility migrates trace links on absolute occurrence objects to create trace links on the latest revision of the absolute occurrence objects.

Note:

This utility is not a part of the automatic upgrade. Run this utility after upgrading the corporate server.

SYNTAX

```
req_migrate_bomview_tracelinks [-u=user-id {-p=password | -pf=password-file} -g=group] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

If used without a value, the system assumes a null value. This argument is mutually exclusive with the **-pf** argument.

If this argument is not used, the system assumes the *user-id* value to be the password.

-pf

Specifies the password file.

The file must be a single-line ASCII file containing the password in clear text. Teamcenter Environment Manager prompts you for a password and creates the password file during installation.

If used without a value, the system assumes a null value. This argument is mutually exclusive with the **-p** argument.

For more information about managing password files, see *Manage password files*.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

You must have modify privileges on all existing trace links in the database to run the migration process. If you do not have modify privileges on some of the trace links, those trace links are not migrated, and a message is written to the log file.

req_migrate_fulltext

Migrates all full text from the input PLM XML file to the Teamcenter database. It can create or update the full text and set the rich text contents of full text.

SYNTAX

```
req_migrate_fulltext [-u=user-id] [-p=password] [-pf=password-file] [-g=group]  
-file=complete-path-to-plmxml-file  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the complete path of input PLM XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.

req_migrate_grm

Migrates all trace links from the input PLM XML file to the Teamcenter database. It creates trace links between the source objects and target objects as specified in the PLM XML file. Trace links that already exist in the database are skipped.

SYNTAX

```
req_migrate_grm [-u=user-id] [-p=password] [-pf=password-file] [-g=group]  
-file=complete-path-to-plmxml-file  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-file

Specifies the complete path of input PLM XML file.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.

req_migrate_richtext

Migrates rich text from Teamcenter 2007 to Teamcenter 8 and later versions. It is required only in special cases to recover rich text from the migrated data that is inaccessible because of format differences.

SYNTAX

```
req_migrate_richtext [-u=user-id] [-p=password| -pf=password-file] [-g=group]  
[-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges.

Note:

If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

For more information about managing password files, see [Manage password files](#).

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

ENVIRONMENT

As specified in *Setting up a Teamcenter command-line environment*.

FILES

As specified in *Log files produced by Teamcenter*.

RESTRICTIONS

To use this utility, you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.