TEAMCENTER

Transition from Teamcenter Enterprise to Teamcenter

Teamcenter 2412

SIEMENS

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

Deploying the Teamcenter Enterprise thin clients

Configuring Teamcenter Enterprise

Configuring Teamcenter

Configuring FMS in Teamcenter

Using Data Exchange from Teamcenter

Using Data Exchange from Teamcenter Enterprise

Standard mapping

1. Getting started with transition to Teamcenter

Teamcenter components required for transition

This guide describes how to install and configure Teamcenter Data Exchange and Teamcenter Enterprise components and provides supported business process descriptions. Data Exchange comprises the following core components:

- Teamcenter (unified architecture) four-tier architecture
- Teamcenter File Management System (FMS)
- Teamcenter Integration Framework
- Teamcenter Enterprise Data Exchange (also referred to as "EMS")
- Teamcenter Enterprise FMS
- Teamcenter Enterprise thin client solutions
- Security Services (if using single sign-on)
- Secure socket later (SSL) communications configured for your Web components and FMS if using HTTPS.

Also, this guide provides procedures for using Data Exchange to transfer data between Teamcenter product lifecycle management sites.

Requirements for transitioning from Teamcenter Enterprise to Teamcenter

Prerequisites	Unless stated otherwise in the description of a configuration task, you must be a Teamcenter **dba** group member or member of another group with **dba** privileges.
Enable Data Exchange	Complete the installation and configuration tasks described in this guide.
Configure Data Exchange	Configuring Data Exchange for Teamcenter Enterprise transition to Teamcenter requires that you configure File Management System (FMS) to communicate between the two sites and set up Teamcenter Integration Framework for

	connecting, mapping, and FMS communications. This is described in the installation and configuration tasks.
Start Data Exchange	Data Exchange is accessible from within the rich client and Teamcenter Enterprise thin client interfaces.

Transition concepts

Following are some basic terms and concepts used when transitioning from Teamcenter Enterprise to Teamcenter.

- **Site**

  A site () is a Teamcenter Enterprise installation or a Teamcenter installation.

- **Export**

  You use Teamcenter data sharing tools for exporting data from a Teamcenter Enterprise or Teamcenter site to remote sites.The data sharing function used to send data to a remote site. While exporting the data, you can either transfer the data with ownership or transfer the data for reference. If you transfer the data for reference, the ownership is retained by the exporting site and a replica is exported to the remote site.

- **Import**

  When you export data from a remote site to a Teamcenter Enterprise or Teamcenter site, the Teamcenter Enterprise or Teamcenter site imports the data automatically.

- **Replica**

  Replication is the act of creating an exact copy of an object, known as a replica, at a specific site. Replicas are objects that are owned by a remote site. Whenever a primary object is modified, you must update the replicas by synchronizing them to the primary object. A replica is a nonwriteable object and cannot be updated except by the owning site. *In Teamcenter Enterprise, you cannot replicate a replica.* When an object is replicated, you cannot delete the primary object unless all the replicas are deleted.

  A replica is represented by a symbol with two green dots.

  In Teamcenter Enterprise and Teamcenter, a replica is represented by a symbol with two green dots.

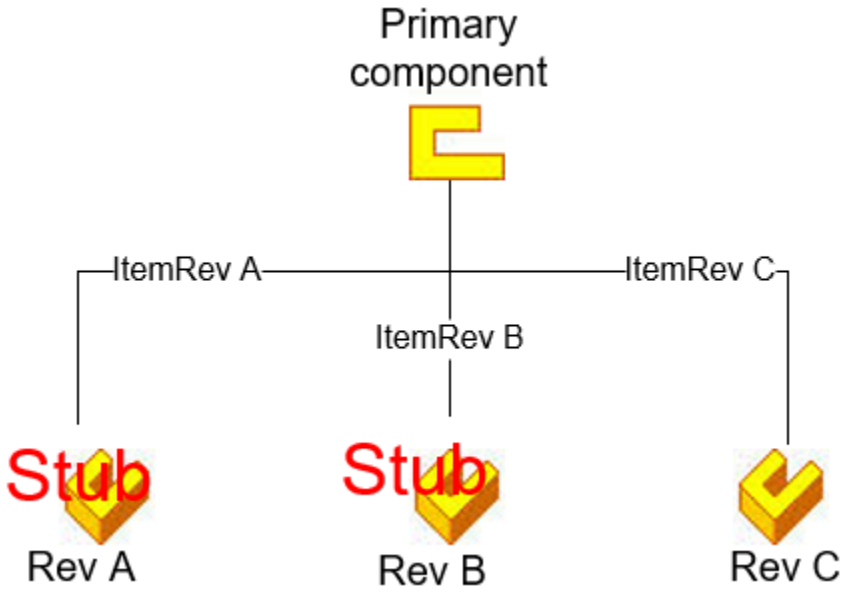
Replica object	Represented in Teamcenter Enterprise	Replica object	Represented in Teamcenter
Assembly		Item	
Component		Item revision	
Document		Document	

- **Stub**

  A stub is a placeholder for an object. Teamcenter Enterprise exports an object as a stub when:

  > Note:
  >
  > The following list applies to data transfers between Teamcenter and Teamcenter Enterprise only.

  - An object has previously been exported to a remote site as a full object, and a subsequent transfer uses the **Include Modified Objects Only** option.
  - The exporting site does not own the object.
  - The object is marked as export protected by a condition.
  - Message access rules deny access to the object.
  - Data swarming is enabled and the exporter spawns new export requests. In this case, objects that are processed in the spawned export requests are exported as stubs in the current export transaction.
  - An inferred delete of relations would occur if the stubs were not included because they are not part of the current transfer option set; for example, if a component with three revisions (**Rev A**, **Rev B**, and **Rev C**) is exported with a transfer option set that specifies the **Latest Revision** export option (**Rev C**), the export content includes the following.



The earlier revisions (**A** and **B**) are sent as stubs so the target site is aware that additional revisions exist. If a subsequent transfer of the component specifies the **All Versions** export option, the stubs at the target site are updated to the full revision **A** and revision **B** objects.

If the first transfer specifies the **All Versions** export option, and a subsequent transfer specifies the **Latest Version** export option, the presence of stubs for revision **A** and revision **B** in the subsequent transfer allows the remote site to know all three revisions still exist and none should be deleted.

- **Synchronization**

  When a primary object is replicated at other sites, you must update the replicas whenever the primary object is modified. The process of updating replicas is referred to as *synchronization*.

  Users can manual synchronize data by re-exporting or re-importing an object using the same transfer formula. Multi-Site Collaboration provides automatic synchronization. Teamcenter Integration Framework supports automatic synchronization for push cases and is initiated by making a call to transfer updated objects.

  Teamcenter Enterprise supports only manual synchronization.

  > Caution:
  >
  > When manually synchronizing a replica, both the owning site and replica site must be online to receive replica deletion notification.

- **Closure rule clause**

A closure rule clause () controls the scope of the data transfer. It defines the rules for gathering objects during the export action. Closure rules specify how the data structure is traversed by specifying the relationships of interest and the actions to occur when these relationships are encountered.

Teamcenter also has many internal closure rules that handle out-of-the-box data model traversal. These are normally imported during installation or upgrade. They must not be changed. For instance, the inclusion of the BOM and Absolute Occurrence network is handled by internal closure rules. In Teamcenter Enterprise, you must be an administrator or member of the **admingrp** group to create a closure rule clause.

- **Transfer mode**

  A transfer mode () is a logical grouping of closure rule clauses. You select a transfer mode when creating closure rule clauses. Transfer modes allow users to export and import data by knowing only the transfer mode name that they must use, for example, **ToSiteA** or **FromSiteB**. In Teamcenter Enterprise, you must be an administrator or member of the **admingrp** group to create transfer mode objects.

- **Transfer option**

  Specifies the different options you can choose to use when Teamcenter transfers an object. The different transfer options are:

  - **Include All Versions**
  - **Latest Versions**
  - **Selected Version**
  - **Include Entire BOM**
  - **Transfer Top Assembly**

- **Transfer option set**

  A transfer option set (TOS) () is a stored set of transfer options used for remote data export. A transfer option set displays all of the unique options in the closure rule conditional clauses for the selected transfer mode. The **Description** column of the **TransferOptionSet** pane in the PLM XML/TC XML Export Import Administration application describes the purpose of the options included in the standard Teamcenter transfer option sets.

- **Factor**

  A factor is a logical concept defined by a set of objects at the exporting site that map to a similar logical concept defined by a different set of objects at the importing site. You define a factor using a closure rule. Closure rules help optimize data sharing performance.

- **File Management System (FMS) file transfer**

  File Management System (FMS) is used for transferring files by geographically separated work groups whose sites are connected by wide area networks (WANs). It also allows access to shared documents in local area network caches, helps avoid round-trips across high-latency WAN networks to reach sites.

  FMS differs from Teamcenter Enterprise replication vaults. Replication pushes files to other locations based on a schedule, whereas FMS pulls files on demand, as users request them. FMS efficiently transfers files across a WAN. FMS can also locate caches closer to end-user machines, for example, FMS server caches (FSCs).

  FMS uses a unique file identifier to determine when to retrieve a file from its local cache rather than from across a network from a remote site. Binary and text files have a different GUIDs if they are replicated. However, if you change the file content by one bit or change its language encoding, the system creates a new file GUID to describe the new contents of the file.

- **Single sign-on (SSO)**

  Security Services allows a user to sign on one time for access to multiple Teamcenter products.

- **BOM-relevant and non-BOM-relevant variants**

  BOM-relevant data and non-BOM-relevant data are two variants in the CATIA data is managed.

  Teamcenter CATIA Manager supports only BOM-relevant variants. Therefore, when a non-BOM-relevant data is transferred from Teamcenter Enterprise to Teamcenter, it must map to BOM-relevant data. A single part in Teamcenter Enterprise may be mapped to multiple parts in Teamcenter during the Teamcenter Enterprise to Teamcenter data transfer and vice versa.

- **Export protection**

  Export protection specifies export-protected objects are shared as stubs. In Teamcenter Enterprise, export protection is accomplished by setting the **ExportProtectedC** condition for objects.

- **Replica deletion**

  The replica deletion capability deletes the export record of an object on the owning site when a replica is deleted on the remote site. When an Teamcenter Enterprise replica is deleted on a remote site, the **UpdMstrObjsReplDel** event is triggered that calls a message on the Teamcenter Enterprise site to delete export records associated with the object.

- **Stub replication**

  The stub replication capability ensures that when a primary object replica is sent to another site, the exporter creates an export record for this new site and tags it as **stubbed**.

- **International Traffic in Arms Regulations (ITAR) license**

Teamcenter provides support for enforcing policies of ITAR (International Traffic in Arms Regulations) to control dissemination of certain types of information through ITAR licenses that you can attach to workspace objects. If an ITAR license exists at the destination site, the **license_id** attribute associated with the workspace object can be imported.

- **Audit monitoring**

  The integration framework provides an audit log of transactions during transfers between Teamcenter Enterprise and Teamcenter that is maintained in the integration framework database. The following transactions are logged:

  - Get objects for ownership transfer.
  - Transfer ownership to site.
  - Update ownership transfer to source site.

- **Organizational/license object audit logs**

  Data sharing uses the TC XML import and export (TIE) functionality to transfer objects between sites. TIE does not support replicating audit record business objects. Do not attempt to replicate the following objects between sites:

  - **Fnd0LicenseChangeAudit**
  - **Fnd0LicenseExportAudit**
  - **Fnd0OrganizationAudit**

- **Activity status**

  The activity status allows you to see the status of each transaction, transfer request, replication request, and synchronization request. Teamcenter displays the activity status in a Web browser when you select the **Show progress indicator** check box in the **Remote using Global Multi-Site** dialog box. This allow you to check the status and monitor in real time the progress of a transfer request. You can also search for a request using specific search criteria supplied through a search dialog box.

- **Swarming**

  Swarming is the process of splitting a larger transaction into smaller sub-transactions and executing them simultaneously. Site administrators specify the maximum number of concurrent transfers that can be initiated from a single export request or many subrequests. Site administrators also specify the approximate number of objects that can be exported per transfer.

- **Switch**

  The **Switch** () class, a subclass of the **Owned Itm** class, is used to group objects for ownership transfer. The objects can be attached to a switch either during a reference transfer or later. Using the

flip-the-swtich ownership transfer functionality, the administrator can transfer the ownership of the objects to the remote site.

- **Flip-the-switch ownership transfer**

  The flip-the-switch ownership transfer provides a mechanism to execute a bulk ownership transfer from Teamcenter Enterprise to Teamcenter in a reasonable time frame.

- **Infer delete**

  Infer delete is a mechanism of deleting relations between objects that were present in the earlier import but not present in the current import. This mechanism informs the importing site about any relation deletion that occurred in the exporting site.

- **Effectivity**

  Effectivities specify the proposed or actual timing of when a change takes effect.

  In the Teamcenter Enterprise CMII model, effectivities state the implementation point of the highest level bill of materials affected by a change, and they can be stated in a variety of ways.

  In Teamcenter, there are two types of effectivities used to identify the valid use of an aspect of product data:

  - *Unit effectivity* specifies the range of item units or serial numbers.
  - *Date effectivity* specifies the range of dates. This is also known as an incorporation point.

  > Caution:
  >
  > Although Teamcenter Enterprise supports alphanumeric (**AplhUnitEffFrom**/**AplhUnitEffTo**) attributes for effectivity, the standard mapping between Teamcenter Enterprise and Teamcenter supports transfer of date (**DateEffectiveFrom**/**DateEffectiveTo**) attributes and numeric (**NumUnitEffFrom**/**NumUnitEffTo**) attributes only for effectivity.

- **As-built structure exchange**

  Data sharing supports transfer of Teamcenter as-built data to an Teamcenter Enterprise Service Lifecycle Management (SLM) system. This is accomplished by exporting the data from Teamcenter and importing it into Teamcenter Enterprise. You can also import as-built data in TC XML format from a third-party system into Teamcenter.

  - Exporting logistics data from Teamcenter SLM As-Built to neutral parts in Teamcenter Enterprise

    When you export an item, Teamcenter exports the part logistics data and maps the data from the item to a neutral part. The part ownership is retained by the owning site unless it is explicitly

transferred. If you are-import the same object, Teamcenter updates the modified information as appropriate.

- Exporting as-built structure from Teamcenter to Teamcenter Enterprise SLM In-Service

  Teamcenter exports an as-built structure from Teamcenter to Teamcenter Enterprise along with the following related data:

  - Physical primary part
  - Physical part revision
  - Primary form (both physical primary part and revision)
  - Physical realization
  - Item and item revision from the physical part is realized
  - Physical part structure
  - Physical part usage
  - Allowed deviation
  - Deviation document authorizing the deviation

  In Teamcenter Enterprise, sharing data creates the physical structure to occurrence relation if the occurrence name is provided. If the occurrence name does not match with the neutral structure occurrence name, the structure relationship is processed without the alignment relation. The item structure information is not exported along with the items.

  The item and item revision information from the physical part is realized and the physical part realization information is exported along with the physical part data. The item is either created or updated depending on standard SLM behavior.

  The physical part and its realized from relation are created between the physical part and neutral structure element. The top-level physical part is created with unrestricted (**World**) access.

  Physical structure is exported along with the occurrence information. If the occurrence name is specified, and an occurrence with the specified name exists in the system, the alignment relation is created. The physical structure information is updated depending on the effective dates determined by comparing the following:

  - The installation date and time for the imported object with the installation and uninstallation date on the physical structure.

  - The time-in and time-out on physical part to location.

  Ownership of imported data is retained by the current owning site unless it is explicitly transferred. If the same object is re-imported, any modified information is updated.

- Importing physical structure from third–party data

  You can import physical structure data into Teamcenter if it conforms to the TC XML schema. The as-built structure is created using the **RealizedProduct**, **RealizedProductRevision**,

Lot, **LotRevision**, **RealizedProductUsage**, **AllowedDeviation**, **Deviation**, and **DeviationRevision** elements. If the occurrence information is provided along with the occurrence name, the alignment relation is created. If the occurrence name is not provided or the occurrence name does not match the neutral part occurrence name, the structure relationship is processed without the alignment relation.

- **Administrative objects**

  The TC XML framework processes Teamcenter administrative objects differently than other objects. For example, if an administrative object identified in the TC XML file is not found at the importing site, the import fails whether or not the exporting user selected the **Continue On Error** option. Objects of the following classes are considered administrative objects by the framework:

  - **ADA_License**
  - **ClosureRule**
  - **Condition**
  - **CondValAgent**
  - **CondValAgentRevision**
  - **CondValData**
  - **CondValDataRevision**
  - **Filter**
  - **Fnd0AlgebraicFormula**
  - **FunctionalityRule**
  - **Group**
  - **GroupMember**
  - **IdContext**
  - **ImanType**
  - **NoteType**
  - **PIEActionRule**
  - **POM_imc**
  - **PropertySet**
  - **PSOccurrenceType**
  - **PSViewType**
  - **ResourcePool**
  - **RevisionRule**
  - **Role**
  - **TransferMode**
  - **TransferOptionSet**
  - **TC_Project**
  - **UnitOfMeasure**
  - **User**
  - **VerificationRule**

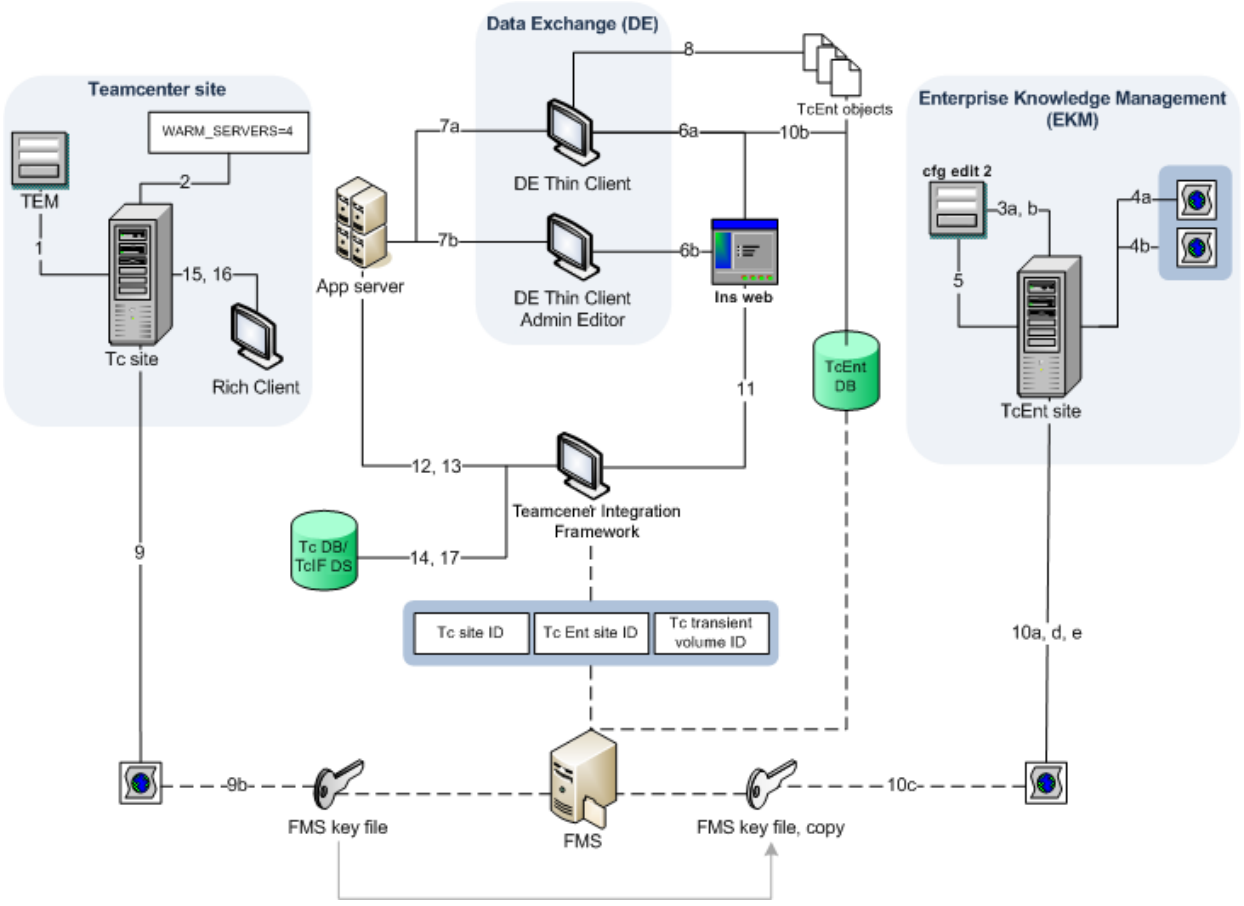
2. Installation process

Task interdependence

Some information provided in this topic is optional depending on the business processes you plan to support. Information about supporting functionality, such as single sign-on (SSO), is described with reference to the general documentation provided for those features. You must follow the defined installation and configuration process due to the product interdependence.

The following diagram shows the installation process activities and where they occur.

> Note:
>
> Dotted lines indicate where information is obtained for a particular task.



1. Install the Teamcenter Integration Framework feature with the Enterprise Integration.

2. Configure the Teamcenter server manager for the minimum number of required **tcserver** processes.

3. Install the Data Exchange solution on the Teamcenter Enterprise server.

    a. Install the EMS module.

    b. Install the required patch.

4. Load the Teamcenter Enterprise Data Exchange data.

    a. Load the administration data.

    b. Load the sample data.

5. Create or edit the Teamcenter Enterprise configuration variables.

6. Use the **insweb** tool to create the Teamcenter Enterprise Web tier Data Exchange solutions.

    a. Create the Data Exchange Web application.

    b. Create the Data Exchange administration editor Web application.

7. Deploy the Teamcenter Enterprise Web tier Data Exchange solutions on your application server.

    a. Deploy the Data Exchange Web application.

    b. Deploy the Data Exchange administration editor Web application.

8. Create the Data Exchange required objects in Teamcenter Enterprise.

    - File system
    - Vault
    - Work location
    - Local federated installation
    - Sites
    - transfer option set

9. Configure FMS on Teamcenter.

    a. Set the **FMS_HOME**, **JAVA_HOME**, **FSC_HOME** environment variables.

    b. Create the encryption key file.

    c. Edit the FMS primary configuration file.

10. Configure FMS on Teamcenter Enterprise.

    a. Set the **FMS_HOME**, **JAVA_HOME**, **FSC_HOME** environment variables in the **pdmsetup** script.

b. Create the **FMS Configuration Details** object.

c. Copy the encryption key file from the Teamcenter site.

d. Edit the FMS primary configuration file to include the required site information.

e. Create the FSC configuration file and startup script.

11. Configure the Teamcenter Integration Framework sites and site mappings using the Teamcenter Integration Framework configuration wizard.

12. Create the Data Exchange sites in Teamcenter.

13. Set the Data Exchange preferences in Teamcenter.

    - **GMS_UID_mapping_dbname_index**
    - **GMS_UID_mapping_remote_site_index**
    - **TCIF_url**
    - **GMS_offline_use_TcGS**

14. (Optional) Install Security Services if you want single sign-on (SSO) functionality.

Installation worksheets

Use these worksheets to record important information about your network. You need to record this information for reference on the worksheet and also get values here when they are needed.

Worksheet 1 – Teamcenter

Item	Value	Default
Server and FMS		
Administrator user name		
Administrator password		
TC_ROOT		
Teamcenter site ID		
Transient volume ID		
KeyGen value		
FMS_HOME directory		
FMS key file name		
fscid		
fscaddress		
Site preference variables (tc_preferences.xml)		
GMS_offline_use_TcGS		**FALSE**
GMS_site_checkin_after_import		**TRUE**
GMS_UID_mapping_dbname_index		
GMS_UID_mapping_remote_site_index		
TCIF_user_nhame		**IFAdmin**
TCIF_user_password		**admin**
TC_Allow_Longer_ID_Name		**FALSE**
TC_gms_export_default_transfermode		**TIEUnconfiguredExportDefault**
TCIF_url		**http://***host-name***:***port-number***/tcif**
TC_gms_import_default_transfermode		**TIEImportDefault**
TC_SSO_login_URL		
TIE_validate_against_xsd		**FALSE**

Worksheet 2 – Teamcenter Enterprise		
Item	**Value**	**Default**
JAVA_HOME		
Server and FMS		
MUX host name		
MUX port number		
DBA user name		
MTI_ROOT directory location		
Teamcenter Enterprise site ID		
Local federated object installation name		
Local federated object installation Id		
FMS_HOME directory		
FMS key file name		
FMS FSC Id		
FMS server URL		
Configuration settings (config.cfg)		
EMS_SSO_ENABLED_LOGIN		
EMS_DEFAULT_STUB_TEAM		
GS_END_POINT		
MONITORING_URL		
ORGANIZATION_NAME		
SCHEMA_VERSION		
XSD_DIR		
XSD_FILE_NAME		
Web tier		
WEB_ROOT directory location		
Web tier application name (context root)		
Teamcenter Enterprise Web application URL		
Data Exchange administrative application URL		

Worksheet 3 – Teamcenter Integration Framework		
Item	**Value**	**Default**
Teamcenter Integration Framework web URL		
Map control files		
Teamcenter Enterprise user name and password		
Teamcenter user name and password.		
fsc.uris		
Teamcenter Integration Framework administrator user name		
Teamcenter Integration Framework administrator password		
vol.id		
fsc.key		

Worksheet 4 – Security Services		
Item	**Value**	**Default**
Security Services logon URL		
Teamcenter Integration Framework administrator user		
Teamcenter site transfer user		
Teamcenter Enterprise site transfer user		
Teamcenter site pseudo site ID (**SSOAppID**)		
Teamcenter Enterprise site pseudosite ID (**SSOAppID**)		

Teamcenter Enterprise requirements

Teamcenter Enterprise Data Exchange components

Teamcenter Enterprise Data Exchange comprises the following core components:

- Teamcenter Enterprise server
- Teamcenter Enterprise user Web application
- Teamcenter Enterprise administration editor Web application
- Teamcenter File Management System

Hardware requirements

Verify whether your Teamcenter Enterprise network meets the hardware requirements for a successful installation of the Data Exchange solution.

For more information about the hardware requirements, see the Teamcenter Enterprise *Network and Database Configuration Guide*.

Software requirements

You must verify your operating system supports the requirements for installing Data Exchange on the enterprise tier and Web tier.

For information about supported platform versions, see the Hardware and Software Certifications knowledge base article on Support Center.

Before you begin installing Data Exchange, ensure that you have installed the following software:

- Teamcenter Enterprise corporate server with Teamcenter Enterprise Foundation solution
- Teamcenter Web Application Manager
- Thin client
- e!Vista
- Latest product patches
- (Optional) Classic client

> Note:
>
> Some Teamcenter Enterprise Data Exchange administrative actions are available only through the classic client. Siemens Digital Industries Software recommends installing the classic client on hosts used by Teamcenter Enterprise Data Exchange administrators.

Reference documentation

Make sure you have the following manuals available for reference when you install Data Exchange:

- The latest Teamcenter Enterprise release bulletin. This document contains the latest information about Data Exchange issues that may affect your installation.
- Teamcenter Enterprise installation manuals:
  - *Teamcenter Enterprise Installation on Windows Server* or *Teamcenter Enterprise Installation on Linux Servers*
  - *Teamcenter Enterprise Client Installation*

  These manuals contain certain procedures that you may need during Data Exchange installation.

If you use other Siemens Digital Industries Software products or third-party software with Data Exchange, you may also need to refer to the documentation for those products.

3. Installing Data Exchange on the enterprise tier

Extract Data Exchange server installation files

To add the Data Exchange server solution on the Teamcenter Enterprise corporate server, and all local area and work group servers, you must copy the installation files from the Teamcenter Enterprise Data Exchange distribution image to your host.

> Note:
>
> If you install the CATIA Metaphase Integration (CMI) solution on a corporate server on which CATIA Automotive Solution (CAS) is already installed, the default mapping functionality for CMI ceases to work. To correct this, you must customize the mapping functionality.

1. Download the Teamcenter Enterprise Data Exchange software install image from the Teamcenter download page for your operating system.

2. Extract the contents of the downloaded file into a staging directory.

3. Download the **EMS_BOOK.EXE** file (Windows) or **EMS_BOOK.TZ** file (Linux) into the staging directory:

4. Extract the contents of the **INSTALL_EMS.EXE** file (Windows) or **INSTALL_EMS.TZ** file (Linux) from the staging directory to the *MTI_ROOT* directory.

5. Stop the MUX and dispatcher if they are running.

Add the Data Exchange server solution

> Note:
>
> You must install the Data Exchange server solution on Teamcenter Enterprise corporate server before applying any maintenance patches (MPs).

1. Launch Configuration Editor (**cfgedit2**).

2. Click **Modify Disk Locations**, click **Add** in the **Disk Locations** dialog box, and add the location of the staging directory.

3. Click **OK** and then click **Add Server Solutions**.

4. In the **Add Server Solutions** dialog box, select the **Teamcenter Data Exchange** server solution and click **OK**.

5. Start the MUX and dispatcher and check the logs for errors.

Pro/ENGINEER integration

Multi-Site support for Pro/ENGINEER Integration (PMX)

For transferring Pro/ENGINEER data, install Multi-Site Support for Pro/ENGINEER Integration (PMX) after adding Teamcenter Data Exchange server and Pro/ENGINEER Classic Integration (PRO) solutions.

The PMX solution customizes the behavior of Teamcenter Data Exchange to suit Pro/ENGINEER data exchange.

Following is the summary of PMX solution:

- Icon definitions for different classes represent different states of objects.
- Class constants with value set definitions dictate the series for export.
- Class constants do not allow Pro/ENGINEER classes as root objects.
- The **BusImPmi** class is inserted immediately under the **BusImEms** class for implementing the following messages:
  - **BusImPmi:FilterOtherSideObjAndRel**

    This message filters the object for relation based closure rule expansion. The message returns an object with revision and sequence same as the revision and sequence of the input object.
  - **Part:GetMultipleRootObjects**

    The input to this message are the **ProPrt**/**ProAsm** objects of the latest version of the document attached to an Advanced Product Configurator (APC) part. The message returns set of **ProPrt**/**ProAsm** objects as the root object.
  - **ProObj:GetPredForOwnershipTransfer**

    If no predecessor is found using the successor relation, use the following steps to find the predecessor:

    ■ Navigate to the business item attached to the current **ProObj** object.

    ■ Get the predecessor of the business item.

- Navigate to the related data item.
- Return the data item as the predecessor.

- **ProObj:GetLatestObjForSeries**

  To navigate to an input **ProObj** object, find the latest version of the attached business item on the left.

Add Multi-Site Support for Pro/Engineer Integration (PMX)

1. Download the Pro/Engineer Integration (PMX) software install image from the Teamcenter Enterprise download page and extract the contents.
2. Launch Configuration Editor (**cfgedit2**).
3. Click **Modify Disk Locations**, click **Add** in the **Disk Locations** dialog box, and add the location of the staging directory.
4. Click **OK** and then click **Add Server Solutions**.
5. In the **Add Server Solutions** dialog box, select the **Multi-Site Support for Pro/Engineer Integration** solution and click **OK**.
6. Start the MUX and dispatcher and check the logs for errors.

CATIA integration

Multi-Site support for CATIA Integration (CMX)

For transferring CATIA data, install Multi-Site Support for CATIA Integration (CMX) after adding Teamcenter Data Exchange server and CATIA Teamcenter Integration (CMI, GMI) solutions.

The CMX solution customizes the behavior of Teamcenter Data Exchange to suit CATIA data exchange.

Following is the summary of CATIA Integration (CMX):

- Icon definitions for different classes represent different states of objects.
- Class constants with value set definitions dictate the series for export.
- The **g0GenBinCmx** class is inserted immediately under the **g0GenBin** class for implementing the following messages:
  - **g0GenBinCmx:g3CheckIfItemInScope**

This implementation is to prevent the loading of Teamcenter (unified architecture) authored data into CATIA.

- **g0GenBinCmx:GetSuccessrForExport**

  If successor part is not found from **AtParent** call, this message tries to get the successor by expanding the **Copied** relation. The copied object is considered as successor only if the corresponding business items have successor relation between them.

Add Multi-Site Support for CATIA Integration (CMX)

1. Download the CATIA Integration (CMX) software install image from the Teamcenter Enterprise download page and extract the contents.
2. Launch Configuration Editor (**cfgedit2**).
3. Click **Modify Disk Locations**, click **Add** in the **Disk Locations** dialog box, and add the location of the staging directory.
4. Click **OK** and then click **Add Server Solutions**.
5. In the **Add Server Solutions** dialog box, select the **Multi-Site Support for CATIA Integration** solution and click **OK**.
6. Start the MUX and dispatcher and check the logs for errors.

Add the Data Exchange solution patch

> Caution:
>
> You must install the Data Exchange Solution to the base foundation before applying any maintenance patches (MPs).

1. Open a command shell and change to the **config** directory under *MTI_ROOT* and type the following commands:

   ```
   pdmsetup
   updatedb -o tmti.bak -n tmti.prd
   ```

2. Launch Configuration Editor (**cfgedit2**).
3. Click **Maintenance Pack Manager** and then click **Download Maintenance Pack**.
4. In the **Download Maintenance Pack** dialog box, select the required patch from the **Maintenance Pack to Download** list, and then click **OK**.

5. Click **Install Maintenance Pack** and then click **OK**.

6. Restart the MUX and dispatcher and check the logs for errors.

Updating local area and work group servers

License manager

If you have changed the license manager configuration on the corporate server, you must verify that the **LM_LICENSE_FILE** system environment variable is set correctly on the local area or work group server before upgrading them.

The **LM_LICENSE_FILE** system environment variable must be set to:

port@host-name

port is the port used by the license manager, and *host-name* is the host on which the corporate server runs.

Copy installation and configuration files

Copy Data Exchange server installation files

Copy the installation files required for Data Exchange on the local area or work group server.

Copy configuration files from the corporate server

Copy configuration files from the corporate server to each local area server and each work group server. Configuration files are in the following directories on the corporate server:

*MTI_ROOT***install\hosts\common**
*MTI_ROOT***install\hosts***target-home-name*

Replace *target-home-name* with the name of the local area or the work group server host. Copy all the files in these directories to the *MTI_ROOT***install** directory on the local area or the work group server.

Run the enterprise setup file to set the environment variables

Run the Teamcenter Enterprise setup file (**pdmsetup**) to set the Teamcenter Enterprise environment variables.

1. Windows system:

   Open a command prompt and enter the following command:

MTI_ROOT\config\pdmsetup.bat

2. UNIX system:

   C shells: source *MTI_ROOT*/config/pdmsetup

   Korn and Bourne shells: *MTI_ROOT*/config/pdmsetup.sh

Run the installation script

1. From the command prompt, run the installation script by typing the following command:

   ```
   esvm insmenu
   ```

2. Answer the installation script prompts. Accept all default paths shown (unless the location of the install images has changed). After you answer all necessary prompts, the installation script displays the following prompt:

   ```
   Teamcenter Enterprise Installation Prompts Completed Previously
   You have made the following selections:
   Installation (MTI_ROOT) directory: path
   Location of the install images: path

   Are your answers correct [Y/N] [Y]?
   ```

3. Type **Y**.

   The installation script installs the Teamcenter Enterprise Data Exchange server solution on the host.

Executing the pushall command

Propagate configuration changes to local area servers by running the **pushall** command on the corporate server.

For information about the **pushall** command, see the appropriate Teamcenter Enterprise server installation manual.

Loading the Teamcenter Enterprise Data Exchange data

Load the emscleanup.dat file

1. Check if the **emscleanup.dat** file is available in the *MTI_ROOT***samples\ems** directory. If yes, perform the remaining steps in this section. If no, then skip this section.

The **emscleanup.dat** file contains the cleanup scripts to delete outdated messages, message groups, message access rules, and user groups.

2. In a command shell, change to the *MTI_ROOT***config** directory.

3. Run the **pdmsetup** script.

4. Change to the *MTI_ROOT***samples\ems** directory.

5. Run the **bldrora** or **bldrmss** command to clean up EMS module data.

   For example, if you are using an Oracle database (*admm70a* represents the name of your administrative database, which is the default value):

   ```
   bldrora -f emscleanup.dat -n admm70a
   ```

Load Data Exchange administrative data

1. Change to the *MTI_ROOT***dbinit** directory.

2. Run the **bldrora** or **bldrmss** command to load the administrative data, for example:

   ```
   bldrora -f emsadmdb.dat -n admm70a
   ```

Data Exchange sample data

The Teamcenter Enterprise Data Exchange solution provides sample data files that are based on a subset of the following standard data models:

> Note:
>
> The closure rules in the sample data files are based on the standard mapping and may not be appropriate if some other mapping is used. Before loading a sample data file, review its contents to determine if it is appropriate for your environment.

Sample data file	Description
apcsampledata.dat	Contains the sample data for Data Exchange based on the Advanced Product Configurator data model.
cmxsampledata.dat	Contains the sample data for Data Exchange based on the CATIA Integration data model.

Sample data file	Description
pmxsampledata.dat	Contains the sample data for Data Exchange based on the Pro/ENGINEER Integration data model.
idssampledata.dat	Contains the sample data for Data Exchange based on the IDS data model.

Load the sample data files that are appropriate for your environment.

Load Data Exchange sample data

1. Change back to the *MTI_ROOT***samples***change_me* directory and make a backup copy of the *xxx***sampledata.dat** file.

   Replace *change_me* with:

   - **ems** for **apcsampledata.dat** and **idssampledata.dat** files.
   - **cmx** for **cmxsampledata.dat** file.
   - **pmx** for **pmxsampledata.dat** file.

   Replace *xxx* with **apc**, **cmx**, **pmx**, or **ids** depending on the data model.

2. Edit the *xxx***sampledata.dat** file as follows:

   a. Remove the **Site** class entries.

      You create the **Site** class entries later in *Data Exchange sites*.

   b. Remove the **TransOpt** class entries.

3. In the command shell, change to the *MTI_ROOT***config** directory and run the **pdmsetup** script.

4. Change to the *MTI_ROOT***samples****ems** directory and run the **bldrora** or **bldrmss** command to load the *xxx***sampledata.dat** file, for example:

   ```
   bldrora -f name_of_dat_file.dat -n admm70a
   ```

4. Creating Teamcenter Enterprise Data Exchange Web applications

Create the Data Exchange thin client solution

1. Launch the Web Application Manager (**insweb**).

2. Click **Copy ICDs**, browse to the **webclient** directory in your staging directory, select the **INSTALL_EMSCLIENT.jar** file, and click **OK**.

3. Extract the required thin client patch sets into your *WEB_ROOT* directory.

4. Click **Copy ICDs**, browse to the following directories in your patch download directory and copy the ICDs:

   *release-version_patch-number***ptier\icd**
   *release-version_patch-number***foundation\icd**
   *release-version_patch-number***admineditor\icd**
   *release-version_patch-number***multisite\icd**

5. In the Web Application Manager, click **Add**.

6. Type the following values:

Name	**ems_user**
Staging Location	*WEB_ROOT***ems_user**

7. Click **Advanced Web Application Options**, change the default **Deployable File Name** box value to **ems_user**, and click **OK**.

8. Click **Modify** and add the paths to the following install image locations:

   WEB_ROOT

   *WEB_ROOT***images**

9. Click **Solutions**, select the following solutions and click **OK**:

   Teamcenter Presentation Tier Infrastructure
   Product Data Management
   Teamcenter Enterprise Data Exchange

10. Modify the context parameters for **MUXHost** and **MUXPort** to match your Teamcenter Enterprise environment and click **OK**.

Create the Data Exchange administration editor solution

1. In the Web Application Manager, click **Add**.

2. Type the following values in the indicated box.

Name	**ems_admin**
Staging Location	*WEB_ROOT***ems_admin**

3. Click **Advanced Web Application Options**, change the default **Deployable File Name** box value to **ems_admin**, and click **OK**.

4. Click **Solutions**, select the following solutions and click **OK**:

   Teamcenter Presentation Tier Infrastructure
   Administration Editor
   Teamcenter Enterprise Data Exchange Administration Editor

5. Modify the context parameters for **MUXHost** and **MUXPort** to match your Teamcenter Enterprise environment and click **OK**.

5. Deploying the Teamcenter Enterprise thin clients

Basic deployment

The Web Application Manager places the deployable file in the **staging** directory. For example, if your *WEB_ROOT* directory is **c:\webbase** and your *Web-application-name* is **ems_user_mp03a**, this file is **c:\webbase\staging\ems_user_mp03a.war**.

These basic deployments procedures provide instructions for deploying the Teamcenter Enterprise Web tier applications (WAR files) in selected configurations on supported servers. Instructions for enabling secure socket layer (SSL) on a supported server are provided in the server documentation.

For information about versions of operating systems, third-party software, Teamcenter software, and system hardware certified for your platform, see the Hardware and Software Certifications knowledge base article on Support Center.

For more information, see the vendor documentation for the following servers:

- Tomcat
- JBoss
- WebSphere

Deploy on the Tomcat server

You can use the Apache Tomcat App Manager to deploy your Teamcenter Web application or you can hot deploy the application by copying the WAR file tin the Tomcat WebApps directory.

Deploy on the JBoss application server

> Note:
>
> This procedure assumes that you installed JBoss using the default installation location and will use the default server location for deploying your Web applications. You must perform this procedure for each Teamcenter Enterprise Web application you want to deploy on the this application server.

1. Copy the Teamcenter Enterprise WAR file (**ems_user_p03a.war** or **ems_admin_p03a.war**) to the following directory under your JBoss installation directory:

   *JBoss-installation***/server/default/deploy**

2. Open the **ear-deployer.xml** file in this directory and ensure the **Isolated** attribute is set to **false**:

```
<attribute name="Isolated">false</attribute>
```

3. Expand the **jboss-web.deployer** subdirectory and open the **server.xml** file in a text editor.

4. Locate the following entry and set the **emptySessionPath** attribute value to **false**:

```
<Connector port="8009" address="${jboss.bind.address}" protocol="AJP/
1.3"
         emptySessionPath="false" enableLookups="false"
redirectPort="8443" />
```

5. Expand the **META-INF** subdirectory, open the **jboss-service.xml** file, locate the following entry, and set the **UseJBossWebLoader** attribute to **true**:

```
<!-- A flag indicating if the JBoss Loader should be used.
     This loader uses a unified class loader as the class
     loader rather than the tomcat specific class loader.
     The default is false to ensure that wars have isolated
     class loading for duplicate jars and jsp files.
-->
<attribute name="UseJBossWebLoader">true</attribute>
```

6. Start the JBoss application server by typing **run –b** *myhost* in a command shell.

> Note:
>
> You must start the application server instance with the bind option to enable connections from clients running on a host different from the application server host. The simplest way to do this is to start the server with the **–b** *myhost* option. Substituting the host name or IP address of the local host for *myhost*. However, this has some security implications.
>
> For information about JBoss security, see the JBoss documentation at **https://www.jboss.org**:

Deploy on the WebSphere application server

This procedure deploys one instance of WebSphere Application Server hosting the Teamcenter Enterprise Web tier application (WAR file). You must perform this procedure for each Teamcenter Enterprise Web application you want to deploy on the this application server.

For more information, see the complete WebSphere documentation available at:

https://www.ibm.com

1. Start the WebSphere integrated solutions console.

   For more information about the console, see the **WebSphere documentation**.

2. In the navigation tree, expand **Applications** and click **Install New Application**.

3. In the **Preparing for the application installation** pane, type the path to, or browse to, the location of the Teamcenter Enterprise Web tier WAR file (**ems_user_p03a.war** or **ems_admin_p03a.war**) in the **Full path** box.

   Select **Prompt me only when additional information is required** and click **Next**.

4. Accept the default **Select installed options** for enterprise applications and modules and click **Next**.

5. In the **Map modules to servers** pane, if you have multiple server instances, select the check boxes for all modules and map them to the same server instance. Click **Next** again.

6. In the summary pane, click **Finish**. Wait for WebSphere to complete the application deployment.

7. Scroll to the top of the page and click **Save**.

   Your application is now deployed and can be started.

6. Configuring Teamcenter Enterprise

Teamcenter Enterprise objects required by Data Exchange

Before you create the administration data specific to Data Exchange, you must have the following in your Teamcenter Enterprise environment:

- A file system for your Teamcenter Enterprise database.
- Vaults, vault locations, and work locations for your Data Exchange objects.
- Teams, team folders, and team locations. You must create one team corresponding to a Data Exchange user. You must create at least one team folder. For every owner in Teamcenter, there has to be a corresponding team or vault in Teamcenter Enterprise. In Teamcenter Enterprise, objects are allowed to be imported only into teams or vaults. If the objects imported into teams have the **DestinationFolderName** attribute populated in the **TCEXML** file, the objects are related to the corresponding folder name.

  > Note:
  >
  > The team objects are required only for IDS integrations.

- For data transfer to work between Teamcenter and Teamcenter Enterprise—for example, a remote export from Teamcenter site to Teamcenter Enterprise site— there must be an owner that has the same name as that of the exporting user at the Teamcenter site.
- **Message access rules** and location selection rules to allow your Data Exchange users to transfer objects to the Data Exchange vaults.

> Note:
>
> For an explanation of how to perform the Teamcenter Enterprise tasks, see the *Teamcenter Enterprise Administrator's Manual*.

Before transferring data between Teamcenter Enterprise and Teamcenter, you must configure the following:

- A *local federated installation* object for use in the FMS configuration.
- *Site* objects for each site participating in transfers.
- A *transfer option set* with the same name as the transfer option set that Teamcenter uses to transfer data to Teamcenter Enterprise.

- *Transfer modes* and *closure rules*.

  > Note:
  >
  > You can also **load the sample** transfer modes, transfer option sets, and closure rules from the sample data file.

- A *switch* for grouping objects to be flipped for ownership transfer.
- A *GMS configuration object* to add configuration variables.

Create a local federated installation

The local federated installation is used to configure FMS for Data Exchange transfers.

1. Log on to Data Exchange Administration Editor as a user with administration privileges.
2. Choose **New→Others** in the left navigation pane, select **Local Federated Installation**, and click **Next**.
3. In the **New Local Federated Installation: Properties** pane:
   a. Type a name for your installation, for example, **DEENT**.
   b. Select the local host name from the **Local Gateway Host** list, for example, **MILLABV1**.
   c. The **Installation Id** value is provided by Teamcenter Enterprise for you. Write the **Installation Id** value on the installation worksheet. This example configuration uses **5857e5e0–40b-11de-bca-000c299a411d**.
   d. Click **Finish**.
   e. Exit Data Exchange Administration Editor and, in your Teamcenter Enterprise command shell, cycle the MUX and dispatcher as follows:

```
dspstop -ra
MUXstop -r
```

Data Exchange sites

Sites required for data exchange

A site () is a Teamcenter Enterprise installation or Teamcenter installation. Your Teamcenter Enterprise installation must have one local or self site and a remote site for each Teamcenter site participating in Data Exchange transfers.

Open a support case on Support Center to get a Teamcenter Enterprise site ID value.

The remote Teamcenter **Site ID** value must match the **fmsenterprise** element's **id** value from your Teamcenter site's FMS primary configuration file. This file is created when you install Teamcenter. Before you create the sites, get this value from the primary configuration file in the *TC_ROOT***fsc** directory and record it in the install worksheet for reference. The file name has the following format:

fmsmaster_FSC_*host-name_user _service-name***.xml**

Create a local site

1. In Data Exchange Administration Editor, choose **New→Others** from the left navigation pane.

   > Note:
   >
   > You can also create a site from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Site**.

2. Select **Site** from the list of classes and click **Next**.

   You can also create a site from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Site**.

3. In the **New Site: Properties** pane, enter the following information:

Box	Description
Site Name	Type a unique name for the site, for example, **DEENT**. Record this value in the installation worksheet.
Site ID	Enter the Teamcenter Enterprise site ID value. Open a support case on Support Center to get this license value. Record it in the install worksheet.
Thin Client URL	Type the URL of the Teamcenter Integration Framework web server used for your Data Exchange transfers, for example, **http://MILLABV2:8080/tcif/controller/index**.

Box	Description
Site Type	Select **Self**.
Database Name	Select a database that is associated with the site, for example, **SU DB**.
Default Vault	Select a vault, for example, **vaulttcif**.
System Type	Type **GMS**.
Vault Location Name	Select a vault location name, for example, **vaulttcif_L**. Note: The vault and vault location names are used for storing the transaction logs and TCE XML files during data transfer.
Ideal Dispersion Rate	This attribute has no effect on a local site.
Preferred Transfer Size	This attribute has no effect on a local site.

4. Click **Finish**.

Create a remote site

Configuration of multiple Teamcenter sites to connect to a Teamcenter Enterprise site is not supported. The user should create only one remote site in Teamcenter Enterprise.

If the GMS setup has multiple Teamcenter sites, the data from Teamcenter Enterprise will be transferred to one Teamcenter site only. If required, this Teamcenter site can transfer the data to other Teamcenter sites.

1. In Data Exchange Administration Editor, choose **Create→Others** from the left navigation pane.

   Note:

   You can also create a site from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Site**.

2. Select **Site** from the list of classes and click **Next**.

3. In the **New Site: Properties** pane, enter the following information:

Box	Description
Site Name	Type a unique name for the site, for example, **DETCUA**. Record this value in the install worksheet.
Site ID	Type the identifier for the remote Teamcenter site, for example, **457706589**. This must match the Teamcenter site ID value that you recorded in the install worksheet.
Thin Client URL	Type the Teamcenter Integration Framework web server URL, for example, **http://MILLABV2:8080/tcif/controller/index**.
Site Type	Select **Remote**.
Default Vault	Type or select the vault name, for example, **vaulttcif**.
Default Vault	Type **GMS**.
Vault Location Name	Type or select the vault location name, for example, **vaultcif_L**. Note: The vault and vault location names are used for storing the transaction logs and TCE XML files during data transfer.
Database Name	Type of select the particular database that is associated with the site. For example, **SU DB**.
Ideal Dispersion Rate	Specifies the maximum number of concurrent export requests that can be initiated within a single export transaction. The default value is 0. To enable swarming, the value must be greater than 0.
Preferred Transfer Size	Specifies the approximate number of objects to be exported per transfer.

4. Click **Finish**.

Transfer option sets in Teamcenter Enterprise

Teamcenter Enterprise transfer option set requirements

A transfer option set (TOS) () is a stored set of transfer options for remote export of data. Only users belonging to the **admingrp** user group can create or modify a TOS.

A **Transfer Ownership** dynamic attribute is added to the transfer option set. This attribute is available when you create or update a transfer option set. This attribute validates whether the TOS can be used for transfer of ownership.

You must have a TOS in Teamcenter Enterprise that matches the name of the TOS your Teamcenter site uses for transferring items. The **TIEOptionSetDefault** TOS is the default TOS for Teamcenter transfers.

Create a transfer option set in Teamcenter Enterprise

1. In Data Exchange Administration Editor, choose **New→Others** from the left navigation pane.

   > Note:
   >
   > You can also create a transfer option set (TOS) from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Transfer Option Set**.

2. Select **Transfer Option Set** from the list of classes and click **Next**.

3. In the **New Transfer Option Set: Properties** pane, enter the following information:

Box	Description
Transfer Option Set Name	Type **TIEOptionSetDefault**.
Transfer Option Set Description	Type **Scope of data to export**.
Site Name	Select the exporting site name, for example, **DE_TCUA**.
Switch Name	Specifies the name of the switch to which the objects transferred using the TOS will be attached in order to do a flip-the-switch ownership transfer later. Note: If you specify a switch name, you must select the **Include Files** and **Include Documents** check boxes.
Transfer Mode	Select **Mode1**.
Version Option	Select **Selected Version**. When creating other option sets, the available options are: • **Include All versions** Transfers all version of the selected object. • **Latest Version**

Box	Description
	Transfers the latest version of the selected object. • **Selected Version** Transfers the version selected by the user. Caution: If you use this option to export an assembly, it must contain precise parts only. If you attempt to export an assembly that contains imprecise parts, alternates, or substitutes, the export fails.
BOM Option	Select **Include Entire BOM**. When creating other options sets, this specifies the options for exporting either the entire product structure or the top-level assembly: • **Include Entire BOM** For a precise case, transfers the entire structure. For an imprecise case, the data scoper exports objects based on a user-defined relation. • **Transfer Top Assembly** Exports only the top-level assembly and not the entire structure. Note: If you selected **Selected Version** as the version option, you must not select **Include Entire BOM** as the BOM option.
Transfer Ownership	Clear (not checked). This is a dynamic attribute and is not stored in the TOS while creating or updating the TOS. Certain attributes, such as **Include Modified Objects Only**, are mutually exclusive with the **Transfer Ownership** session option. This attribute checks whether the current set of options are compatible with the transfer ownership option.
Include Modified Objects Only	Clear (not checked).

Box	Description
	If selected, only objects that have been modified since the last transfer are exported. Caution: If you select the **Include Modified Objects Only** check box, you must not select **Transfer Top Assembly** as the BOM option. If data in Teamcenter is replicated to a remote site: • You cannot delete the replicated objects and synchronize them to remote sites using the **Include Modified Objects Only** option. You must export the full object set to delete the replicated objects. • You cannot synchronize **license_id** objects or structure effectivity using the **Include Modified Objects Only** option. You must export the full object set to update this information.
Include Files	Select (checked). Includes all underlying operating system files (such as JT, drawing files) with each dataset selected for import or export.
Include Documents	Select (checked). Includes document objects in the export. For a part-centric data model, this allows the user to select, at transfer time, to transfer documents for a part-centric data model.

4. Click **Finish**.

Create a transfer mode

You select a transfer mode when creating closure rule clauses. You must be a member of the **admin grp** group to create or modify transfer mode objects.

1. In Data Exchange Administration Editor, choose **Create→Others** from the left navigation pane.

> Note:
>
> You can also create a transfer mode from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Transfer Mode**.

2. Select **Transfer Mode** from the list of classes and click **Next**.

3. In the **New Transfer Mode: Properties** pane, enter the following information:

Box	Description
Transfer Mode Name	Specifies the name of the transfer mode object, for example, **Mode1**.
Transfer Mode Description	Specifies the description of the transfer mode, for example, **Part-JT**.

4. Click **Finish**.

Closure rule clauses

Create a closure rule clause

1. In Data Exchange Administration Editor, choose **Create→Others** from the left navigation pane.

> Note:
>
> You can also create a closure rule clause from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Closure Rule Clause**.

2. Select **Closure Rule Clause** from the list of classes and click **Next**.

3. In the **New Closure Rule Clause** pane, enter the following information:

Box	Description
Closure Rule Clause Name	Specifies the name of the closure rule clause.
Transfer Option	Specifies the transfer option for which the closure rule is used.
Class1	Specifies the class name of the object to which the closure rule is applied.
Class2	Specifies the class name of the object that must be gathered using the closure rule.

Box	Description
Relationship Name	Specifies the relationship from **Class1** to **Class2**. Note: You must specify either the relationship name or the foreign keys on **Class1** and **Class2**.
Foreign Key on Class1	Specifies the attribute on **Class1**, which is pointed by an attribute on **Class2**.
Foreign Key on Class2	Specifies the attribute on **Class2**, which is pointed by an attribute on **Class1**.
Primary Object	Helps to export the appropriate objects when you select the **Include Modified Objects Only** option in the transfer option set. In this case, the primary object is sent as a full object and the secondary object is sent as either full object or stub.
Relation Type	Specifies whether the closure rule is used for finding the spannable/nonspannable objects/relations. You can choose between **Spannable**, and **NonSpannable**. Objects gathered using spannable or nonspannable relations are written in the TCE XML file Spannable relation connects objects that span across different islands. Nonspannable relation groups objects in the same island.
Applicable for Stubs`	Specifies whether the closure rule must be used even if the **Class1** object is exported as a stub.
Skip Secondary Objects Validation	If set to **true**, specifies that only the primary object must be validated to determine whether the factor is modified or not. This attribute is applicable only while transferring objects that are modified since the last export.
Factor Group Number	Groups closure rules. All closure rules with the same factor group number are processed together. There can be only one primary object for such closure rules.
For Root Objects Only	Specifies that the closure rule can be applied only for root objects.
Filter Class2 Object	Specifies that the objects of **Class2** for a factor are filtered using a specific rule. By default, only the latest object is fetched. The filtering mechanism can be customized using the **FilterOtherSideObjAndRel** message.

Note:

- Closure rules without an associated transfer option are considered to be the default closure rules and are applicable irrespective of what options the user selects in the transfer options and session options.

4. Click **Finish**.

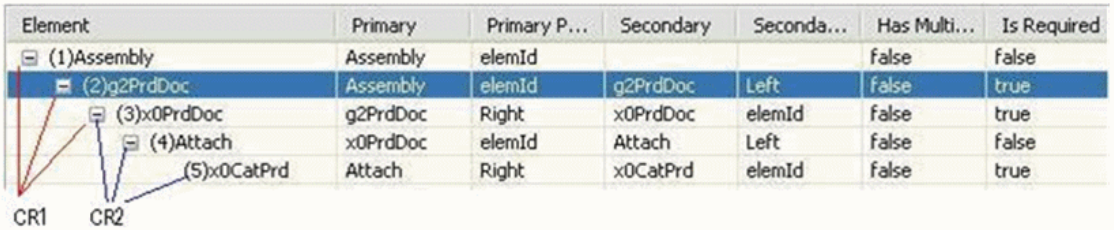
Closure rule and factors

To transfer data from an Teamcenter Enterprise site to a Teamcenter site, the Teamcenter Integration Framework data mapper maps an object's attributes and its relationships to other objects in Teamcenter Enterprise to the representation in Teamcenter. The data mapper divides the input TCE XML from the Teamcenter Enterprise site into smaller chunks called factors.

The data mapper identifies the objects that should belong to a factor using filtering rules. The Teamcenter Enterprise administrator must configure closure rules that align with the mapper filter rules for a factor. The configuration of closure rules does not play any role in transferring objects that the user selected. However, when the user wants to transfer only the objects that are modified since the last export, the factor-based configuration of closure rule plays a vital role. All the closure rules that contribute to a single factor are grouped together in a factor group. The **Factor Group Number** field of a closure rule clause specifies to which factor group the closure rule belongs to.

The simplest configuration is one factor group containing one closure rule that results in three objects (left object, relation object, and right object). These three objects belong to a single factor. A factor group can have only one primary object. All other objects are considered as secondary objects. While transferring objects that are modified since last transfer, the primary object is transferred as a full object if one of the secondary object or the primary object itself is modified since the last transfer.

For example:



In this example, **Assembly** is the primary object. Both the product document (**x0PrdDoc**) and CATIA product (**x0CatPrd**) connected to a given assembly are grouped as a single factor and mapped to the objects at the Teamcenter site. To support this factor, the Teamcenter Enterprise administrator creates the following two closure rules.

	Class1	Relation	Class2	Factor group number	Primary object
CR1	**Assembly**	**g2PrdDoc**	**x0PrdDoc**	CATIA assembly factor	**Class1**
CR2	**x0PrdDoc**	**Attach**	**x0CatPrd**	CATIA assembly factor	

Both closure rules are grouped in a single factor group by specifying the same factor group number.

- **Case 1**

  Consider that **x0CatPrd** alone was modified since last transfer. In the subsequent transfer that includes only the modified objects:

  - **Assembly** is transferred as full object.
  - **x0PrdDoc** is transferred as stub.
  - **x0CatPrd** is transferred as full object.

- **Case 2**

  Consider that **Assembly** alone was modified since last transfer. In the subsequent transfer that includes only the modified objects:

  - **Assembly** is transferred as full object.
  - **x0PrdDoc** is transferred as stub.
  - **x0CatPrd** is not transferred.

Closure rule and islands

An island is a unit of objects that Teamcenter importer imports. An Teamcenter Enterprise exporter does not know what constitutes an island in Teamcenter, an Teamcenter Enterprise site administrator defines closure rules that gather objects to be grouped in an island that maps to an island in Teamcenter. All objects in an island are either ownership transferred or reference transferred. One Teamcenter Enterprise island can split and result into multiple Teamcenter islands.

While defining a closure rule, if the Teamcenter Enterprise administrator specifies the relation type as nonspannable, all objects and relations gathered using this closure rule are grouped in a single island. However, if the relation type is specified as spannable:

- All the objects gathered using this closure rule will belong to different islands.

- All the relations gathered will belong to the island to which the left object of the relation belongs.

The Teamcenter Enterprise exporter provides the continue on error feature, which allows the user to specify whether to continue the export in case of recoverable errors. The continue on error feature is supported only for Teamcenter Enterprise to Teamcenter data transfers.

- **Behavior when the continue on error option is set**

  - While gathering any island, if the Teamcenter Enterprise exporter encounters any recoverable error, it provides stub representation only for those objects that are required for data integrity. The exporter then traverses beyond these stubbed objects to process other islands.

  - During ownership transfer, if any object in an island cannot be transferred for ownership, the Teamcenter Enterprise exporter downgrades the intent for all the objects of the island to reference transfer and continues with the processing of other islands.

  - During import, if Teamcenter importer cannot accept an object, all the objects in that island are rolled back and the referenced objects are stubbed. In case of ownership transfer, if the imported cannot accept ownership of any specific island, the ownership transfer for all the objects in the failed island and its dependent islands are not performed. Stubs are created for item and item revision of the failed islands.

    All the objects available in the failed island are added to the failed objects XML file and importer log.

- **Behavior when the continue on error option is not set**

  - While gathering an island, if an object in the island fails, the export action is aborted.

  - During ownership transfer, if any object in an island cannot be transferred for ownership, the export action is aborted.

- **Behavior when transferring islands containing immutable objects**

  Objects marked as immutable are not modifiable and therefore these object must be analyzed carefully. The standard (OOTB) behavior is immutable objects included in an island prevents the ownership transfer of the island. This causes an export of the assembly for ownership transfer to fail.

- **Behavior when a transfer request's root object is an AssmMstr object**

  When you select an **AssmMstr** object as the root object of a transfer, the exporter substitutes the assembly object as the root object. Therefore, you must account for this when determining where your closure rule navigation must start.

Sample closure rule with Include Entire BOM as transfer option

The following closure rule is used when the user selects a transfer option set with BOM option as **Include Entire BOM**. If you want to use this closure rule irrespective of the options in the transfer option set, do not enter any value in the **Transfer Option** box.

Note:

If you select the transfer option as **Include Entire BOM** while creating the closure rule and the transfer option set has BOM option as **Transfer Top Level Assembly**, the closure rule is used to gather objects that are sent as stubs (also referred as not-in-formula objects) to maintain the data integrity.

Box	Values
Closure Rule Clause Name	**AssmStrc**
Transfer Option	**Include Entire BOM**
Class1	**Assembly**
Class2	**AssmMstr**
Relationship Name	**PartMstrsInAssembly**
Foreign Key on Class1	
Foreign Key on Class2	
Primary Object	**Class1**
Relation Type	**Spannable**

Sample closure rule for Applicable for Stubs

In the following closure rule, the **Applicable for Stubs** box is set as **True**. This closure rule is processed even when the object of **Class1** is written as a stub in the TCE XML file. This closure rule is used when the mapper requires information from more objects to create an equivalent stub object for the other system. For example, the **AssmMstr** object is always written along with the **Assembly** object even if the **Assembly** object is written as a stub.

Box	Values
Closure Rule Clause Name	**ItemRev_1**
Transfer Option	
Class1	**Assembly**
Class2	**AssmMstr**
Relationship Name	**ItemMstrsForStrucBIRev**
Foreign Key on Class1	
Foreign Key on Class2	

Box	Values
Primary Object	**Class2**
Relation Type	**NonSpannable**
Applicable for Stubs	**True**

Sample closure rule with primary object set to Class1

In the following closure rule, **Class1** is an assembly and **Class2** is a document. The assembly and document are related using a **PartDoc** relation. **Class1** is set as the primary object. If you update the assembly and want to transfer only the modified objects, the assembly is sent as a full object and the document is sent as a stub. If you update the document and want to transfer only the modified objects, both the assembly and document are transferred as full objects.

Box	Values
Closure Rule Clause Name	**PartDoc_2**
Class1	**Assembly**
Class2	**Document**
Relationship Name	**DocumentDescribingPart**
Foreign Key on Class1	
Foreign Key on Class2	
Primary Object	**Class1**
Relation Type	**NonSpannable**

Switch

Flip the switch ownership transfer

A switch () is used for grouping objects that need to be flipped for ownership transfer. The administrator creates a switch and attaches it to a *transfer option set (TOS)* by specifying the name of the switch in the TOS.

While exporting the objects as replicas to a remote site, the user selects a TOS that has a switch. Later, the **Flip The Switch** action transfers the ownership of objects that are already transferred as replicas to the remote site. Before performing this action, you must ensure that all these objects are attached to a switch.

You can perform the following actions on a switch:

- *Attach an object to a switch*.
- *Detach an object from a switch*.
- *Get objects in a switch*.

The administrator performs the following actions on a switch:

- *Perform dry run for ownership transfer*.
- *Flip ownership transfer*.
- *Get objects in a switch*.

Create a switch

1. In Data Exchange Administration Editor, choose **New→Others** from the left navigation pane.

   Note:

   You can also create a switch from classic client. To do so, choose **Create→Administrative Classes→GMS Classes→Switch**.

2. Select **Switch** from the list of classes and click **Next**.

3. In the **New Switch: Properties** pane, enter the following information:

Box	Description
Switch Name	Specifies the name of the switch. You specify the switch name while creating a transfer option set in order to group objects for ownership transfer.
Site Name	Specifies the name of the destination site to which you want to transfer the ownership of the objects once the switch is flipped successfully.
Switch Description	Provides basic information about the switch.

   Note:

   Switch States and **Flip Transaction ID** are two attributes of a switch that are generated and maintained by the system. You can view these attributes while updating or querying for a switch.

   The following are the different switch states:

- **None** — Indicates the switch is not yet flipped. This is the default state of the switch when it is created. In this state, you can add objects to or remove objects from the switch.
- **Switch Flip Executing** — Indicates that the **Flip The Switch** action is currently being executed for the switch. In this state, you cannot add objects to or remove objects from the switch.
- **Switch Flip Succeeded** — Indicates that the **Flip The Switch** action is successfully executed for the switch. In this state, you cannot add objects to or remove objects from the switch. Also, you cannot flip the switch again.
- **Switch Flip Failed** — Indicates that the **Flip The Switch** action failed to execute successfully for the switch. In this state, you can add objects to or remove objects from the switch. You can flip the switch again. This state is similar to the **None** state, except that the **Flip The Switch** action was executed on the switch previously.

The **Flip Transaction ID** box specifies the transaction ID of the **Flip The Switch** action. You can use this transaction ID to track the export transaction in the Teamcenter Integration Framework configuration user interface.

4. Click **Finish**.

GMS configuration

Initial GMS configuration

As part of the Data Exchange installation process, the necessary configuration variables were set for you in the **config.cfg** file. As an administrator, you were able to modify or add to the configuration variables using the Teamcenter Enterprise Configuration Editor. Now, you can add or update some of the configuration variables using the GMS configuration object. This object contains the different configuration variables required by Teamcenter Enterprise Data Exchange. To create a GMS configuration object, see *Create GMS configuration*.

To see the list of configuration variables that can be updated only in **config.cfg** file, see *Additional configuration variables*.

Create GMS configuration

1. In Data Exchange Administration Editor, choose **New→Others→GMS Configuration** and click **Next**.

2. In the **GMS Configuration** pane, enter the required values and click **Finish**.

Note:

You can also create a GMS configuration object in classic client. To do so, choose **Create→Administrative Classes→GMS Classes→GMS Configuration**.

The following table lists and describes the different configuration variables available in the GMS configuration object:

Box	Description	Variable
Teamcenter Global Services URL	Specifies the middleware end point URL.	**GS_END_POINT**
Teamcenter Global Services User Name	Specifies the user name Teamcenter Enterprise uses to log on to the Teamcenter middleware component (Teamcenter Integration Framework or Global Services).	**EMS_GS_USERNAME**
Teamcenter Global Services User Password	Specifies the password associated with the user name Teamcenter Enterprise uses to log on to the Teamcenter middleware component.	**EMS_GS_USERPW**
Enable SSO for GMS	Specifies whether SSO is enabled or disabled for Teamcenter Enterprise when the user is logged on to the Teamcenter middleware component. By default, the value is set to **FALSE**. You must set the value to **TRUE** to enable SSO logon.	**EMS_SSO_ENABLED_LOGIN**
Teamcenter Global Services SSO Application ID	Specifies the application ID of Teamcenter middleware component in Security Services.	**GS_APPID**
Debug Level	Specifies the server method debugging level. By default, the value is set to **NONE**. The error log is sent to Teamcenter middleware component log file. You must use the GMS configuration user interface to access the contents of the log file.	**EMSDEBUG**
Switch Debug Level To High On Error	Controls the behavior of the debug levels in case of errors. By default, the value is **FALSE**. If you set the value to **TRUE**, the value of the debug level is automatically switched to **High** when an error is encountered.	**SWITCH_EMSDEBUG_-TO_HIGH_ON_ERR**

Box	Description	Variable
Schema File Name	Specifies the name of the XML Schema Definition (XSD) file that is used by the importer to validate the XML file to be imported.	**XSD_FILE_NAME**
Directory Path For Schema File	Specifies the directory name from the corporate server where the XML Schema Definition (XSD) file is generated. The generate XML schema action must be executed on the corporate server or from a client connected to the corporate server. This location path must be a full path to a directory or a folder that is writable by the trusted user, using the appropriate syntax for the platform.	**XSD_DIR**
Transaction Retry Count	Specifies the numbers of times a Data Exchange transaction must be retried.	**EMS_TRANS_-RETRY_COUNT**
Export System Attributes	Specifies whether to write the system attributes in the XML file during export. By default, the value is **TRUE**.	**SKIP_SYSTEM_ATTRIBUTE**
Delete Temporary Files	Specifies whether to delete the temporary files that are generated during the export and import operations. By default, the value is **TRUE**.	**EMS_TMP_FILE_-DELETE_DISABLE**
Exporter Cache Size	Specified the number of objects cached in the memory during an export transaction. Additional objects are spilled into the database. This is a performance and a scalability tuning parameter that is based on the hardware and software configuration. Setting a very high value may affect the scalability by limiting the number of concurrent servers. Setting a very small value may degrade the performance by increasing the database I/O. By default, the value is set to 10000.	**EXPORTER_CACHE_SIZE**
Exporter Spill Database Name	Specifies the name of the database to be used for temporarily spilling objects from the memory cache during an export transaction.	**EXPORTER_TEMP_-DB_NAME**

Box	Description	Variable
GSID Cache Size	Specifies the memory cache size for GSID that is used while import. This is a performance and a scalability tuning parameter that is based on the hardware and software configuration. Setting a very high value may affect the scalability by limiting the number of concurrent servers. Setting a very small value may degrade the performance by increasing the database I/O. By default, the value is set to 1000.	**EMS_GSIDCACHE_SIZE**
Batch Size For Comparing GS Identity Objects	Specifies the batch size for comparing GS identity objects during import. This is a performance and scalability tuning parameter based on the hardware. Setting a high value may affect the scalability whereas setting a very small value may degrade the performance. By default, the value is set to 400.	**MAX_IDENTITY_-OBJECTS_TO_COMPARE**
Batch Size For Inserting GS Identity Objects	Specifies the batch size for writing GS identity objects during import. This is a performance and scalability tuning parameter based on the hardware and software configuration. Setting a very high value may affect the scalability by limiting the number of concurrent servers. Setting a very small value may degrade the performance by increasing the database I/O. By default, the value is set to 10000.	**MAX_IDENTITY_-OBJECTS_TO_INSERT**
Fetch Size For Retrieving GS Identity Objects	Specifies the batch size for retrieving GS identity objects during import. This is a performance and scalability tuning parameter based on the hardware and software configuration. Setting a very high value may affect the scalability by limiting the number of concurrent servers. Setting a very small value may degrade the performance by increasing the database I/O. By default, the value is set to 10000.	**MAX_IDENTITY_-OBJECTS_TO_RETRIEVE**

3. Click **Finish**.

Additional configuration variables

Variable	Description
DESC_SEPARATOR	Specifies the separator character between the transfer option set (TOS) name and description. The default value is a **,** (comma).
EMSGSSERVICESMAP	Specifies the map used to construct certain services URLs related to Global Services. The default value is: **LoginService:** **/WebServices/BOSService** **DataTransferService:** **/controller/invoke**
EMS_SITE_ALLOWS_-TRANSFER_ON_REIMPORT	Specifies whether an object must be imported to the vault or the team to which it was previously imported to. To import the object to a different vault or team, set this value to **true**. If you do not specify a value for this configuration variable or set the value to **false**, the object will be imported to the same vault or team to which it was previously imported. For example, consider that a part was earlier imported by Teamcenter Enterprise to Team A. The owner of this part is changed to Team B in the remote site. Now, if you want the part to be imported to Team B, set the value of the **EMS_SITE_ALLOWS_TRANSFER_ON_REIMPORT** configuration variable to **true**. If the variable is not set to **true**, the object is imported to Team A.
ENABLE_EMS_OPERATIONS	Specifies whether Data Exchange is enabled in Team Browser when replica behavior is also enabled (using **ENABLE_REPLICA_BEHAVIOR**). The default value is **1**, enabling Data Exchange if replica behavior is also enabled. To disable Data Exchange in Team Browser, set **ENABLE_EMS_OPERATIONS** to **0** or set **ENABLE_REPLICA_BEHAVIOR** to **0**.
ENABLE_REPLICA_BEHAVIOR	Specifies whether replica behavior is enabled in Team Browser. This is a subset of overall Data Exchange functionality, and is required for use of the complete Data Exchange feature set. The default value is **1**, enabling replica behavior. To disable Data Exchange in Team Browser, set **ENABLE_REPLICA_BEHAVIOR** to **0**.
FMS_HOME	Specifies the FMS home directory. For server side, **FMS_HOME** specifies the home directory for FSC files, and on the client machine, **FMS_HOME** specifies the home directory for FCC files.

Variable	Description
ROOT_OBJECT_XSD_FILE	Specifies the name of the XSD file against which the XML format of the root object is validated.
SSL_CERT_DIR	Specifies the path to the trusted single socket layer certificates file.
SSL_CERT_FILE	Specifies the file name of the trusted single socket layer certificates file in PEM format.
VALIDATE_XML_FOR_IMPORT	Specifies that XSD validations must be performed on the XML file during import.

7. Configuring Teamcenter

Install Teamcenter Integration Framework

Install Teamcenter Integration Framework. See the *Teamcenter Integration Framework* documentation.

Configure server manager

Data Exchange requires a minimum of four warm (started) **tcserver** processes.

1. Open the **serverPool.properties** file for your Teamcenter server.

2. Change the **PROCESS_WARM** value to minimum value of **4** and save the file.

   The **PROCESS_WARM** pool-specific parameter sets the desired number of prestarted but unassigned Teamcenter servers.

3. Restart the server manager.

Configure sites in Teamcenter

Sites are created and configured using the Organization application. You must configure the existing Teamcenter site for Data Exchange and create a site object for the Teamcenter Enterprise site.

You must define sites for all locations that exchange data.

1. In the rich client, open the Organization application.

2. Expand the top-level sites node  from the **Organization List** tree.

3. Select the existing site, select the existing test in the **Site Name** box, and type the name the site is known as in Data Exchange, for example, **DETCUA**.

4. Select the following check boxes:

   Uses TcXML Payload
   HTTP Enabled
   Allow deletion of replicated master objects to this site

5. Type the URL for your Teamcenter Integration Framework web server (**http://MILLABV2:80811/tcif**) in the **TcGS URL** box.

6. Click **Modify**.

7. Select the top-level **Sites** node from the **Organization List** tree.

8. Type **DEENT** in the **Site Name** box.

9. Type the unique nine-digit identifier for the Teamcenter Enterprise site in the **Site ID** box. Get the value you recorded in the install worksheet (**923756038** for this example). Do not use leading zeros in the identifier.

10. Select the following check boxes:

    Uses TcXML Payload
    HTTP Enabled
    Allow deletion of replicated master objects to this site

11. Type the URL for your Teamcenter Integration Framework web server (**http://MILLABV2:8081/tcif/controller/index**) in the **TcGS URL** box.

12. Click **Create**.

Set Teamcenter preferences

You must set several preferences for Teamcenter Integration Framework in Teamcenter. Some of these may have been set as part of the process of installing the Teamcenter Integration Framework. However, you may not have had the required information at that time.

1. Log on to the rich client as user with Teamcenter administrator privileges.

2. Choose **Edit→Options** and click **Filters**.

3. Type **GMS_UID** in the **Search by preference name** box, click **Edit**, and set the following preferences:

    ```
    GMS_UID_mapping_dbname_index=MILLAB:sum70a
    GMS_UID_mapping_remote_site_index=MILLAB:0
    ```

    The **dbname** value consists of the first six characters of the Teamcenter Enterprise host name, a colon, and the Teamcenter Enterprise user database name. The **remote_site** value is the first six characters of Teamcenter Enterprise host name, a colon, and an index value, usually **0**.

Set version 0 transfer option set

Teamcenter stores two versions of a dataset by default (version 0 and version latest; version 0 is a copy of latest version). When exporting an item or item revision that contains an **IMAN_specification** relation to a dataset, Teamcenter exports two datasets pointing to the same **ImanFile** object. For online transfers to Teamcenter Enterprise, you must export the version 0 only. You can change the export behavior to

export only version 0 of the dataset with the latest version exported as a stub. To do this, you must add the **opt_from_tc_ds0_only** option to your transfer option set and set its value to **True**.

> Caution:
>
> Do not set this option to **TRUE** when you are transferring objects through a briefcase file. If this option is not set to **FALSE**, the briefcase import fails.

1. In the Teamcenter rich client, open the PLM XML/TC XML Export Import Administration application and select **TransferOptionSet** in the option pane beneath the tree pane.
2. Select the name of the transfer option set you want to use when transferring item or item revisions with related datasets.
3. Click + to add a new option.
4. Select the first column (**Option**) of the new row (at the bottom of the list) and type **opt_from_tc_ds0_only** in the **Option** box.
5. Press the Tab key and type **Zero version export** in the **Display Name** box.
6. Press the Tab key and select **True** from the **Default Value** list.
7. Press the Tab key and type **Export zero version dataset only** in the **Description** box.
8. Press tab and type **Dataset Options** in the **Group Name** box.
9. Click **Create**.

Transfer option sets in Teamcenter

A **Transfer Ownership** dynamic attribute is available when you create or update a transfer option set (TOS). This attribute validates whether the TOS can be used for transfer of ownership.

You must have a TOS in Teamcenter Enterprise that matches the name of the TOS your Teamcenter site uses for transferring items. The **TIEOptionSetDefault** TOS is the default TOS for Teamcenter transfers.

A transfer option set is defined as a local TOS (for export transfers) or a remote TOS (for import transfers). When exchanging data with Teamcenter Enterprise, you must create a remote TOS that matches the name of a TOS defined at the Teamcenter Enterprise site. Data Exchange uses this TOS to pull (import) data from the remote system. The TOS has no associated transfer mode, because the transfer mode (TM) and closure rules are defined at the remote site. The remote site TOS is used to populate the export Teamcenter TC XML file.

Closure rules are filtered based on the options in the remote TOS.

Use the PLM XML/TC XML Export Import Administration application to create a transfer option set.

> Note:
>
> A remote TOS is normally imported from a remote site to ensure that it matches the TOS on the remote site. Although you can create a remote site TOS and manually synchronize it to the remote site, Siemens Digital Industries Software recommends that you use the import (**tcxml_import**) and export (**tcxml_export**) utilities instead to provide your remote TOS. Once a TOS is imported, you must not change the TOS name or site reference; however, you can add or remove options as long as the TOS at the remote site is updated to reflect the changes you make.

For remote transfer option sets, you define only the TOS name, which must match the TOS name defined in the remote system.

You must define a local transfer option set for exporting data to another site. A local TOS requires that you define an option row in the options table for each symbol that is defined in the associated closure rule clause.

> Note:
>
> The **opt_traverse_replica** option must be set to **false** if you are exporting a replica from Teamcenter to Teamcenter Enterprise.

Define users, groups, and rules

The following setup is required to ensure that the Teamcenter Enterprise owner attribute is mapped to Teamcenter owning user and owning group.

1. In Teamcenter Enterprise, create user teams, shared teams, and vaults.

2. In Teamcenter, create a user for every team and vault in Teamcenter Enterprise. General users can also be created for testing mapping between the sites.

3. In Teamcenter, create a group for every team and vault in Teamcenter Enterprise with the corresponding users created previously members with this as their default group.

   This configuration ensures the following standard mapping does not cause an error on import.

From Teamcenter Enterprise	To Teamcenter
User team	Owning user and owning user’s default group
WPS team	Owning user and owning user’s default group
Vault	Owning user and owning user’s default group

From Teamcenter	To Teamcenter Enterprise
Owning user	Owner (team or vault)
Owning group	Not mapped (data not transferred)

Transferring data between sites

Data Exchange provides standard transfer modes that allow you to move standard Teamcenter and Teamcenter Enterprise data between sites. If these standard transfer modes do not meet your requirements, you must provide a transfer mode for each destination site and for each data type you support for transfer to the site. This is true for any type of PLM site, including other Teamcenter sites. Therefore, it is likely that you will have multiple transfer modes for each site. The transfer modes for Data Exchange must navigate clauses in the resulting XML file not the Teamcenter database. The transfer modes defined for a specific site typically uses the same closure rule but have different filters, action rule, or revision rules depending on the type of data being transferred.

Use the PLM XML/TC XML Export Import Administration application to create transfer modes.

Be aware of the following items when working with transfer modes:

- Before performing any TC XML exports, you must remove any references to obsolete attributes and classes from custom closure rules or property sets. Also remove any deprecated attributes and classes from custom closure rules or property sets.

  To identify deprecated and obsolete classes and attributes, see the Teamcenter **README** file for the current version.

- If you are using the CATIA integration and are transferring assemblies with CAD data, you must add the following property set clause to the transfer mode you use to transfer the assemblies:

  ```
  CLASS.POM_catia_absocc:ATTRIBUTE.IsContext:DO
  ```

  If you do not add this clause, the transfer fails during import with an error message stating the export failed at the data import step, and the log file stating the **IsContext** attribute of the **POM_catia_absocc** class is empty.

- The **forceExportReplica** option must be set to **TRUE** only when you are transferring non BOM data. For BOM relevant data, this option should not be used or, if set, it must be set to **FALSE**.

8. Configuring FMS in Teamcenter

FMS tickets and keys

Data Exchange uses File Management System (FMS) keys to digitally sign and verify FMS tickets during file transfers. FMS keys can use different keys for different ticket types; however, this is not a best practice for Data Exchange. Teamcenter Enterprise does not provide a default key value. Because you use the same key for all systems that exchange files, you must either use the Teamcenter FMS default key value or generate a key and use the **install_encryptionkeys** utility to set the key in the Teamcenter database. FMS provides a script (**keygen**) that runs a Java utility for generating a key. This utility requires that you have the **FSC_HOME**, **FSC_HOME**, and **JAVA_HOME** environment variables set.

Data Exchange creates tickets with the **delete** flag set to false to allow a paused transaction to be resumed. Therefore, you must periodically delete completed transactions from the transient volumes.

The examples in this procedure are based on a Teamcenter installation on the **MILLABV2** host, **detcua** service name, and **DETCUA** configuration ID.

Set required environment variables

You must set the following environment variables on your Teamcenter host:

FMS_HOME=*TC_ROOT***\fcc**
FSC_HOME=*TC_ROOT***\fsc**
JAVA_HOME=*TC_ROOT***\install\install\jre**

Create an FMS key

1. Open a Teamcenter command shell:

   Windows systems:

   - Click **Start** and choose **All Programs→Teamcenter** *x.x*→**detcua_DETCUA Command Prompt**.

     Note:

     The **detcua_DTECUA** value is determined by the *service-name* and *configuration-ID* is set during the Teamcenter installation process.

   Linux systems:

   - Open a command shell, change to the *TC_ROOT***/tc_menu** directory, and run the *service-name_configuration-ID***.sh** script.

2. Use the **keygen** script to generate a 128-bit key and create the **fms.key** file that contains it.

   ```
   keygen 128>fms.key
   ```

3. List the contents of the **fms.key** file and copy the key value. You can add comments to the file, however, the key value must be the first non-UNIX style comment line in the file, for example:

   ```
   more fms.key
   # This file (fms.key) contains a 128-bit key for FMS tickets
   f6968bd55d740f44ebacaba09c1313c4
   ```

4. Install the generated key in the Teamcenter database using the **install_encryptionkeys** utility. Type the following command and type **y** when prompted:

   ```
   install_encryptionkeys -u=tc-admin-user -p=password -g=group -f=modify
   ```

   When prompted for the key, paste the copied value and press Enter.

5. Verify the generated key is installed using the following commands:

   ```
   install_encryptionkeys -u=tc-admin-user -p=password -g=group -f=list
   f6968bd55d740f44ebacaba09c1313c4
   ```

   The commands output must match the key value in the **FMS.key** file:

   ```
   more fms.key
   f6968bd55d740f44ebacaba09c1313c4
   ```

6. Verify FMS access for your Teamcenter site using the **fscadmin** utility:

   ```
   fscadmin -s URL-for-FSC -k common-key-file ./status
   ```

   In this example, type the following command:

   ```
   fscadmin -s http://MILLABV2:4544 -k FMS.key ./status
   ```

7. Copy the **fms.key** file to the *MTI_ROOT***fms** (*FMS_HOME*) directory of your Teamcenter Enterprise host.

Edit the FMS primary configuration file

You must edit the FMS primary configuration file for each site to reference the key files that you create and to configure the FSCs from different sites to communicate with each other. The **mulitsiteimport** element is used to set the attributes for each site in the Data Exchange network. In this example configuration for a Data Exchange network there is one Teamcenter site (ID=**457706589**) and one Teamcenter Enterprise site (ID=**923756038**). You use the **Installation Id** value of the **Local Federated**

Installation object you created in the **multisiteimport** element. You recorded this value in the installation worksheet.

You can configure FMS in several ways to obtain different performance and failure tolerance characteristics.

1. Back up the FMS primary configuration file. The file is located in the *TC_ROOT***fsc** directory, and the name has the following format:

   fmsmaster_FSC_*host-name_user_service-name***.xml**

   In this example, back up and edit the **fmsmaster_FSC_MILLABV2_ntpirv_detcua.xml** file.

2. Add the following ticket elements after the **fmsenterprise** element and before the **fscdefaults** element:

```
:
<fmsenterprise id="457706589">
     <!--- Added encryption related tags -->
  <ticket version="v100" keyfile="fms.key" />
  <ticket version="M050" keyfile="fms.key" />
  <ticket version="F100" keyfile="fms.key" />
     <fccdefaults>
:
```

3. Add the following **multisiteimport** element after the **fmsworld** element and before the **fmsenterprise** element:

```
:
<fmsworld>
     <!--- DEENT site installation Id from the Federated Installation
Object) -->
     <multisiteimport siteid="5857e5e0–40b-11de-bca-000c299a411d" >
    <defaultfscimport fscid="FSC_MILLABV1_ntpriv_DEENT"
    fscaddress="http://MILLABV1:8544" priority="0" />
  </multisiteimport>
     <fmsenterprise id="457706589">
:
```

4. Locate the **transientvolume** element and record the **id** value in the installation worksheet. You need this later when you configure Teamcenter Integration Framework.

   > Tip:
   >
   > You may also want to copy the value to a storage document to prevent typing errors.

5. Restart the Teamcenter services.

6. Review the **FSC stdout** and **stderr** log files for errors. These files are located in the *FSC_HOME*directory and are named **FSC_***host-name_user_service-name***stdout.log** and **FSC_***host-name_user_service-name***stderr.log**. In this example, they are:

   FSC_MILLABV2_ntpriv_detcuastdout.log
   FSC_MILLABV2_ntpriv_detcuastderr.log

   Also check the *FMS_HOME***\log\FSC\process\fsc.log** and *FMS_HOME***log\MLD\process\plmconsole.txt** files for errors.

7. Start a Teamcenter command shell and change to the *FSC_HOME* directory.

8. Reload the FSC configuration and check the FSC status using the following commands:

   ```
   fscadmin –s http://MILLABV2:4544 –k fms.key ./config/reload

   fscadmin –s http://MILLABV2:4544 –k fms.key ./status
   ```

   Verify the command returns a message similar to:

   ```
   FSC id: FSC_MILLABV2_ntpriv_detcua, site:457706589
   running for 1 days, 23 hours, 33 min, 49 sec
   Current number of file connections: 0
   Current number of admin connections: 1
   ```

9. Configuring FMS in Teamcenter Enterprise

Modify the pdmsetup script

You must ensure that **JAVA_HOME**, **JRE_HOME**, and **FMS_HOME** are set in the **pdmsetup** script (**pdmsetup.bat** or **pdmsetup.sh**) in your *MTI_ROOT***Config** directory.

The examples in the configuring FMS on Teamcenter Enterprise procedures are based on a Teamcenter Enterprise installation performed by the **ntpriv** user on the **MILLABV1** host, with **8544** as the FMS port number, and **DEENT** as the site name.

For more information about using FMS with Teamcenter Enterprise, see the Teamcenter Enterprise *Network and Database Configuration Guide*.

1. Back up copy the **pdmsetup** script in the **config** directory of *MTI_ROOT*.

2. Open the script in a text editor and check that these values exist below the **MTI_ROOT** entry and point to valid locations, for example:

   Window systems:

   ```
   Set MTI_ROOT=F:\tce7
   Set FMS_HOME=%MTI_ROOT%\fms
   Set FSC_HOME=%FMS_HOME%
   if not defined JAVA_HOME set JAVA_HOME=C:\apps\java\jre1.5.0_11-b03
   Set PATH=%MTI_ROOT%\bin;%MTI_ROOT%\install;%FMS_HOME%\lib;
   %JAVAHOME%\bin\server;%PATH%
   ```

   Linux systems:

   ```
   Set MTI_ROOT=opt/tce7
   Set FMS_HOME=$MTI_ROOT/fms
   Set FSC_HOME=$FMS_HOME
   if [ ! "$ JAVA_HOME " ];
   then set JAVA_HOME=opt/apps/java/jre1.5.0_11-b03
   fi
   Set PATH=$MTI_ROOT/bin;$MTI_ROOT/install;$FMS_HOME/lib;
   $JAVA_HOME/bin/server;$PATH
   ```

   Modify these values as required.

3. Add the following line:

Windows systems:

```
call MTI_ROOT\bin\fscsetup.bat
```

Linux systems:

```
call MTI_ROOT/bin/fscsetup.sh
```

4. Save the script.

5. Run the script in the command shell where your system (MUX and dispatcher) runs.

Create FMS configuration details

> Caution:
>
> You can have only one **FMS Configuration Details** object in your Teamcenter Enterprise site. If is already created, you must either change the **FMS Encryption Key** value to the value set in the **fms.key** files or set the key value in the **fms.key** files and Teamcenter database to this value.

1. In the Data Exchange Administration Editor, choose **New→Others**, select the **FMS Configuration Details** option, and click **Next**.

2. In the **New FMS Configuration Details: Properties** pane, create an FMS configuration with the following values and click **OK**.

Property	Description
File Server Cache URLs	Type the set of server cache URLs for the FMS server locations that have FMS caches, for example, **http:/MILLABV1:8544**.
Is FMS Enabled?	Select (checked).
FMS Encryption Key	Type (or copy and paste) the value from **fms.key** file. You should also be able to obtain this value from the install worksheet.
Ticked Expiration Interval (minutes)	Leave the default value (**1440** minutes or one day).

3. Choose **System Utilities→Update Rules Cache** and click **OK**.

4. Choose **System Utilities→Apply Rules Cache to Environment** and click **OK**.

Configuring FMS files

FMS primary configuration file

The FMS primary configuration file is used to manage the FMS configuration. This file contains the FMS server cache (FSC) group definitions and group-to-group links. The FMS server configuration file is installed for each server. This file identifies the server configuration within the system and optionally overrides FSC and FMS client cache (FCC) parameter defaults specified in the FMS primary configuration file.

Create an FMS primary configuration file

1. Open a command shell, change to the **config** directory of *MTI_ROOT*, and run the **pdmsetup** script.

2. Change to the *FMS_HOME* directory.

3. Generate the FMS primary configuration file by typing the following command:

> Tip:
>
> The **-o** (host) argument is case sensitive. If the host is not found, no error message appears and you may generate an FMS primary configuration file with missing content.

```
fms-cfg-gen -o MILLABV1 -f fmsmaster_FSC_MILLABV1_ntpriv_deent.xml
```

Edit the FMS master configuration file

1. Back up the generated FMS master file (**fmsmaster_FSC_MILLABV1_ntpriv_deent.xml**) you generated in *Edit the FMS master configuration file*.

2. In a text editor, open the **fmsmaster_FSC_MILLABV1_ntpriv_deent.xml** file and add a **multisiteimport** element following the **fmsworld** element as follows:

```
:
<fmsworld>
      <!-- Section of multi-site import tags -->
    <!-- DETCUA -->
    <multisiteimport siteid="457706589">
        <defaultfscimport fscid="FSC_MILLABV2_ntpriv_detcua"
                fscaddress="http://MILLABV2:4544" priority="0" />
    </multisiteimport>
:
```

3. Add or edit the **ticket** elements following the **fmsenterprise** element as follows:

```
:
<fmsenterprise id="5857e5e0-40b4-11de-bca4-000c299a411d ">
      <!--- Required FMS key encryption tags -->
      <ticket version="v100" keyfile="fms.key" />
   <ticket version="M050" keyfile="fms.key" />
   <ticket version="F100" keyfile="fms.key" />
:
```

4. Add empty **fscdefaults** and **fccdefaults** elements, in that order, following the **ticket** elements as follows:

```
:
<ticket version="F100" keyfile="fms.key" />
<fscdefaults/>
<fccdefaults/>
:
```

5. Replace the **change_me** values for the **fscgroup**, **fsc**, and **clientmap** elements with appropriate values, for example:

```
:
fscgroup id="MILLABV1_grp">
         <fsc id="FSC_MILLABV1_ntpriv_deent"
address="http://MILLABV1:8544" ismaster="true">
             <accesson id="tgdtq3iMILLABsum70a--adi"
              root="F:/tce7/FS1/vaultloc"
              username="ntpriv"
         <clientmap subnet="127.0.0.1" mask="0.0.0.0">
            <assignedfsc fscid="FSC_MILLABV1_ntpriv_deent"
priority="0" />
          </clientmap>
       </fscgroup>
:
```

6. Comment out the **linkparameters** element (no need to change the **change_me** values) as follows

```
:
</fscgroup>
<!--
    <linkparameters fromgroup="change_me" togroup="change_me"
           transport="wan" maxpipes="8" />
-->
    </fmsenterprise>
  </fmsworld>
:
```

7. Save the file.

Create an FSC server configuration file

The **FMS_HOME** directory contains the **fsc.xml.template** file you can use the basis for your FSC file.

1. Copy the **fsc.xml.template** file to **FSC_MILLABV1_ntpriv_deent.xml**.

2. Change the **fsc** element's **id** attribute value to **FSC_MILLABV1_ntpriv_deent.xml**

```
<fscmaster serves="true" />

<fsc id="MUXHost"/>
```

3. Save the file.

Modify the startfsc script

1. Edit the **startfsc.bat** or **startfsc.sh** script to reference all configuration adjustments, for example:

```
set fsc_cmd="%JAVA_HOME%\bin\java" -server -Xms%FSC_MEM%
-Xmx%FSC_MEM%
-classpath "%FSC_CP%" "-Djava.library.path=%FMS_HOME%\lib"
-Dfms.config=fmsmaster_FSC_MILLABV1_ntpriv_deent.xml
-Dfsc.config=FSC_MILLABV1_ntpriv_deent.xml
-Dsun.net.client.defaultConnectTimeout=90000
-Dsun.net.client.defaultReadTimeout=90000 %*
com.teamcenter.fms.servercache.FMSServerCache
title FMSServerCacheA FSC_MILLABV1_ntpriv_deent
%fsc_cmd%
```

2. Save the file.

Verify the FMS configuration

1. On Windows systems, change your Teamcenter Enterprise *FMS_HOME***\lib** directory and a rename the **FSCClientProxyv80.dll** file to **FSCClientProxv.dll** and the **FSCClientProxyv80.lib** file to **FSCClientProxy.lib**.

2. Run the **startfsc** script.

3. Check the *FMS_HOME***\log\FSC\process\fsc.log** and *FMS_HOME***\log\MLDC\process\plmconsole.log** files for errors.

4. Type the following commands to validate the FSC can communicate with both systems:

```
fscadmin -s http://MILLABV1:8544 -k fms.key ./status
fscadmin -s http://MILLABV2:4544 -k fms.key ./status
```

You get a message from both FSCs similar to:

```
FSC id: FSC_MILLABV1_ntpriv_deent, site: 5857e5e0-40b4-11de-
bca4-000c299a411d
  running for 0 days, 0 hours, 7 min, 59 sec
Current number of file connections: 0
Current number of admin connections:  1
```

5. In your Teamcenter environment, change to the *FMS_HOME* directory and type the following command to validate the FSC can communicate with both systems:

```
fscadmin -s http://MILLABV2:4544 -k fms.key ./status
fscadmin -s http://MILLABV1:8544 -k fms.key ./status
```

You get a message from both FSCs similar to that shown in the previous step. Ensure the **fmsmaster.xml** file has **accesson** attributes values to relate the team locations and vault locations.

10. Controlling Teamcenter Enterprise Data Exchange behavior

Configuring class constant for data export

You can determine whether or not data belonging to a particular class can be exported by setting the **SelectedAsRootObjectC** class constant.

The following example shows this constant set to prevent export of data items belonging to the **ProData** class:

```
ProData.SelectedAsRootObjectC = "FALSE";
```

Configuring class constant to allow relation manipulation

You can manipulate (create, update, or delete) all relations when the right object of a relation object is a replica. However, when the left object of a relation object is a replica, you control relation manipulation by using the **RelationCanLeftSideBeRemoteC** class constant. By default, the value of this class constant is **-** and you cannot manipulate a relation when the left object is a replica. If you want to allow relation manipulation when the left object is a replica, set the **RelationCanLeftSideBeRemoteC** class constant value to **+**.

The default value of the **RelationCanLeftSideBeRemoteC** class constant is set to **+** for the following relations (including their children):

- **CADOccR**
- **ExRecRel**
- **GAnotate**
- **GenDeriv**
- **OccCfgR**
- **PartDocM**
- **PeAbSdoc**
- **PelmDoc**
- **PrcHsR**

- **RelImpRc**
- **Sig**
- **StubRel**

Configuring class constant for exporting dynamic attributes of an object

To write dynamic attributes in the schema file or to transfer the dynamic attributes along with the objects, the administrator specifies the dynamic attribute names in an attribute list.

> Note:
>
> - Any dynamic attribute to be written in the schema file or to be exported must be added to the attribute list of the class to which it is attached.
> - For any subclass, the dynamic attributes list also includes the attributes mentioned for all the parent classes in the class hierarchy.

The administrator then sets the **EMSExportDynamicAttrListC** class constant to refer to the attribute list. Before writing the dynamic attributes in an XML file, the dynamic attributes must be inflated using the **InflateDynamicAttrExportPre** method.

As an example, the **StrBIApc** class is a subclass of the **WorkItem** class. The value of the class constant for the **WorkItem** class is **WorkItemDynamicAttrList** and the value for the **StrBIApc** class is **StrBIApcEMSDynamicAttrList**.

The attribute list for the **WorkItem** class and **StrBIApc** class are as follows:

```
define attribute list WorkItemDynamicAttrList
with attributes DestinationFolderName;
WorkItem.EMSExportDynamicAttrListC = WorkItemDynamicAttrList

define attribute list StrBIApcEMSDynamicAttrList
with attributes ViewNetwork,
StartingViewName,
ViewInViewNetwork,
EligibleViewSet;
StrBIApc.EMSExportDynamicAttrListC = StrBIApcEMSDynamicAttrList;
```

Apart from the attributes defined for the **StrBIApc** class, the attribute list of this class will also include the **DestinationFolderName** attribute, which is the attribute of the parent class (**WorkItem**).

Setting up nonstructured items to be exported as structured items

For a nonstructured item, the **sub-label** attribute of the **GSIdentity** class identifies the series to which the item belongs. The value of the **sub-label** attribute is not populated during Data Exchange installation to optimize the performance. To export the nonstructured items as structured items, you can populate the value of this attribute after the installation is complete and before the application is launched.

To populate the value of the **sub-label** attribute for a specific class, the user must set the **RelationsForItemC** class constant to a value set containing the relations that define the series.

The following example shows how to populate the **sub-label** attribute for the document object:

```
define value set EMSRelationsForDocument["ReviseR","ChekInR",
"ChekOutR","CkiTeamR","Versn"];
Document.RelationsForItemC = "EmsRelationsForDocument";
```

Append this code in the **.prd** file that is available in the **\meta\ems** folder and then rebuild the model.

Updating an ancestor replica

An object created in Teamcenter Enterprise and transferred to Teamcenter for ownership, becomes an ancestor replica in Teamcenter Enterprise. If this object is modified in Teamcenter and transferred to Teamcenter Enterprise for reference, only selective attributes of the object are updated in Teamcenter Enterprise.

Out of the box, the following selective attributes are updated:

- **DestinationFolderName**
- **DateEffectiveFrom**
- **DateEffectiveTo**
- **FindNumber**
- **Givelist**
- **LifeCycleState**
- **NumUnitEffFrom**
- **NumUnitEffTo**
- **OwnerName**

- **PartNumber**

If you want to add more attributes to the selective list of attributes, you must implement the **GetAttrLstToUpdOnAncstrRepl** published message for the appropriate class and add the attributes to the **attrList** output list.

To add the **LifeCycleState** attribute for the **Cmponent** class, implement the **GetAttrLstToUpdOnAncstrRepl** published message as follows:

```
message Cmponent:GetAttrLstToUpdOnAncstrRepl(
            input:  ObjectPtr      objToImport,
                    ObjectPtr      localObj ::
            output: SetOfStrings  *attrList,
                    integer       *mfail)code
{
            /*Call AtParent message */
            if (setAddStr(*attrList, LifeCycleStateAttr) < 0)
            {
               dstat = uiOutOfMemory();
               goto EXIT;
            }
}
```

11. Mapping data models between Teamcenter Enterprise and Teamcenter

You must create transformation rules for exchanging data between your Teamcenter Enterprise and Teamcenter data models. Use the the Diverse Schema Utility to create mapping rules to transform data during the exchange.

You can generate a schema for your Teamcenter Enterprise data model.

12. Validate subsystem operation

You can now validate that all Data Exchange components are configured properly.

1. For your Teamcenter Enterprise site:

    a. Start the MUX and dispatcher.

    b. Start the FSC.

    c. Start the application server where the Data Exchange thin client solutions are deployed.

2. For your Teamcenter site:

    a. Start the Teamcenter services (these may be started automatically)

    - FSC service
    - Secure file management service

    b. Start the server manager (server pool).

    c. Start the web server (typically the same as Teamcenter Enterprise thin client solutions).

    d. Start Teamcenter Integration Framework.

3. Validate subsystem operation:

    a. Log on to Teamcenter Integration Framework and choose **Configure→Data Store**.

    b. Expand the **bos** folder.

    This validates JDBC connector.

    c. Choose **View→ Activity Status**, expand query pane and click **Search**.

    Ensure Teamcenter Integration Framework does not report an error. This validates the communications with datastore.

    d. Choose **View →BODs**, click **TcEnt -GenericDocument _***siteID* in the **BODs** pane, and click **Continue** in the right query pane.

    Ensure Teamcenter Integration Framework does not report an error. This validates the Teamcenter Enterprise MUX and dispatcher are running and Teamcenter Integration Framework can log on to Teamcenter Enterprise.

e. Click **TeamcenterItem_***siteID* in the **BODs** pane, and click **Continue** in the right query pane.

Ensure Teamcenter Integration Framework does not report an error. This validates the **tcserver** process and server manager are running and Teamcenter Integration Framework can log on to Teamcenter.

f. In an Teamcenter Enterprise command shell, run the following commands:

```
fscadmin -s http://millabv1:8544 -k fms.key ./status
fscadmin -s http://millabv2:4544 -k fms.key ./status
```

Ensure that utility reports that both FSCs are running. This validates the FSCs can communicate from the Teamcenter Enterprise site.

g. In a Teamcenter command shell, run the same commands as in the previous step.

This validates that the FSCs can communicate from the Teamcenter site.

13. Optional Teamcenter Enterprise configuration

Modifying an existing installation to remove unused modules

Teamcenter Enterprise module interdependencies

In the current scenario, when you install Data Exchange on Teamcenter Enterprise, you are forced to install other Teamcenter Enterprise modules, some of which you may not require. Different types of interdependencies for modules in Teamcenter Enterprise force you to install the unnecessary modules.

The following are the different types of interdependencies for modules:

- One module has a run-time dependency on the information contained in another module. As an example, consider that the XYZ module has a run-time dependency on the information available in the ABC module. This dependency can be due to one of the following reasons:
  - The XYZ module implements messages that are defined in the ABC module.
  - The XYZ module defines messages and implements them on classes defined in the ABC module.
  - The XYZ module sets or gets attributes defined in the ABC module.
- A solution must include the functionality that is included in a particular module. For example, most solutions are impaired if the LCM module is not included.

You can remove the unnecessary modules after upgrading to the current level of Teamcenter Enterprise. To remove a module, you must:

- Back up the existing database.
- Back up the existing **tmti.prd** file.
- Remove the administrative data related to the module.
- Regenerate a new rule file.
- Create a new **tmti.prd** file.
- Install the new libraries.
- (Optional) Remove any unused tables in the database.

- (Optional)Remove any unreferenced module libraries.

Database backup requirement

As a precaution against possible data loss and in case the site wants to revert to the previous installation, you must back up all the database files.

The **tmti.prd** file contains the information about the current MODeL configuration. If a site wants to completely remove the database tables that are no longer used, the new MODeL information must be compared to the old MODeL information to find that tables to be removed. Therefore, you must copy the existing **tmti.prd** file to a backup location.

Removing the administrative data related to the module

Administrative data that is common to the rules subsystem, such as message access rules and conditions, generally refer to messages, classes, and attributes that are module specific. During installation, the *XXX***admdb.dat** file is loaded into the database for each *XXX* module that is installed. When a module is removed, the messages, classes, and attributes referred to are no longer defined, thereby leading to run-time errors in the rules subsystem. Therefore, the first step in removing a module is removing the administrative data.

Each module that is no longer required for Data Exchange on Teamcenter Enterprise has a *XXX***admdbcleanup.dat** file. This file contains the delete statements for each object added by the *XXX***admdb.dat** file of the module.

You apply the *XXX***admdbcleanup.dat** file to the current system using the **bldrora** command.

```
bldora -e -n pwf-file -f XXXadmdbcleanup.dat
```

You can use a script file to delete multiple modules. For example, to delete the administrative data for all modules that are not required by Data Exchange on Teamcenter Enterprise, add the following command to a script file:

```
foreach fname (c9s c9t pup tkt vpd mci emg edt vms xrf cds vis dmm wsm)
bldora -e 100 -n pwf-file_path -f
%MTI_ROOT%\support\fname\fnameadmdbcleanup.dat
end
```

Create a new tmti.prd file

1. Run the **rulefile** command to regenerate a new rule file that does not refer to the removed administrative data.

2. Open the **tmti.prd** file for edit and remove the references to the removed modules.

3. Regenerate the **tmti.prd** file.

4. Install the new library files.

Removing unused tables in the database

Any database tables from the modules that you removed are still defined, and possible populated in the database. As the installation no longer refers to these tables, to reduce the database size, you must remove these unused tables. To remove the unused tables, run the **updatedb** utility.

```
updatedb -p pwf_file_for_the_current_database -o name_of_saved_tmti.prd_file -n
tmti.prd
```

The **updatedb** utility does not automatically run commands to modify the existing data. Therefore, the commands in the **alterdb.msql** file must be applied manually to each database in the installation. You must examine the **alterdb.msql** file in a text editor to verify if the file contains proper commands.

The contents of the **alterdb.msql** file is similar to the following:

```
.
.
drop table C9BITPVW ;
drop table C9CFGCT ;
drop table C9SRCPVW ;
drop table C9TRNDEP ;
drop table C9TRNTSK ;
drop table C9TSLEVQ ;
drop table C9VISPVW ;
drop table CAADTDOC ;
drop table CAADTFIL ;
drop table CAAPDATA ;
.
.
```

The tables listed must correspond to the classes defined in the modules that are removed. The **alterdb.msql** file is applied to the databases using **sqlplus**:

```
sqlplus dbname/dbpassword@alterdb.msql
```

Removing unreferenced module libraries

To reduce the installed footprint, you must delete or archive the libraries of the removed modules. You can find the **libsXXX.so** library and the **XXXserv** stand alone server in the *MTI_ROOT***/bin** directory.

Single sign-on (SSO)

Configuring Security Services

Security Services provides a mechanism for accessing various applications without requiring you to log on to each one separately (single sign-on).

After you install Security Services, you must configure the single sign-on (SSO) credentials for your users that are authorized to use Data Exchange. You can configure Team Browser for *single sign-on logon* to support Teamcenter Enterprise Data Exchange operations.

Configure SSO for Data Exchange

1. Configure single sign-on (SSO) for Teamcenter Enterprise.

2. Update the following attributes of the *GMS configuration object*:

Attribute name	Description
Teamcenter SSO Application ID	Specifies the application ID of Teamcenter Integration Framework in Security Services.
Enable SSO for GMS	Specifies whether SSO is enabled or disabled for Teamcenter Enterprise when logged on to Teamcenter Integration Framework. By default, the value is set to **FALSE**. You must set the value to **TRUE** to enable SSO logon.

3. Log on to Teamcenter Integration Framework, choose **Configure→Security**, click the **SSO** tab, and select the **SSO Enabled** ☑ check box.

4. Provide appropriate values the following:

   - **SSO Application ID**

     This must match the ID for Teamcenter Integration Framework set in Security Services Application Registry table.

   - **TcSS Login Redirect URL**

     This must match the URL provided in the Security Services for the client redirect to the logon web page. For Teamcenter, this value is identified in the **TC_SSO_LOGIN_PAGE** environment variable.

   - **TcSS Identity URL**

This must match the URL of the Security Services Identify Service. For Teamcenter, this value is identified in the **TC_SSO_SERVICE** environment variable.

Update the other values if necessary.

Configure SSO for Team Browser

1. Add the following values to the *I-deas Installation***\bin\tb.cmd** file:

```
set SSO_LOGIN_URL = http:\\localhost:8080\ssologin
set SSO_IDENTITY_URL = http:\\localhost:8080\ssoidentity
set SSO_APPLICATION_ID = TCEnterprise
```

> Note:
>
> The **SSO_LOGIN_URL** specifies the location of the Web application that contains the Single Sign-On Login Service. Set this parameter if you use Teamcenter Security Services. Ensure that this value matches the value you specified in the **SSO_SERVICE_URL** configuration variable on the corporate server.

2. Copy the **teamcenter_sso_common.jar** and **teamcenter_sso_applib.jar** files from the Teamcenter Enterprise software distribution image to the **tb\jar** directory in your I-deas installation.

3. Open the **teambrowser.bat** file from the **tb** directory of your I-deas installation and add the following variables to the end of the **CLASSPATH** variable:

```
;%SDRC_TB_LOCAL%\jar\teamcenter_sso_applib.jar;%SDRC_TB_LOCAL%
\jar\teamcenter_sso_common.jar
```

Example:

```
set CLASSPATH=%CLASSPATH%;%TBCLASSES%;.......%CORBACLASSES%
;%SDRC_TB_LOCAL%\jar\teamcenter_sso_applib.jar;%SDRC_TB_LOCAL%
\jar\teamcenter_sso_common.jar
```

4. Add the following entries in the set **ARGS** command:

   - **-ssoLogonUrl %SSO_LOGIN_URL%**
   - **-ssoIdentityUrl %SSO_IDENTITY_URL%**
   - **-ssoAppId %SSO_APPLICATION_ID%**

Optimizing performance of Teamcenter Enterprise Data Exchange

Optimize the search performance of Data Exchange

1. Add the following lines at the end of the *MTI_ROOT***meta\ids\pitids.prd** file:

```
attach index RltLeftIndex to OrntSync;
attach index RltLeftIndex to OccCfgR;
attach index RltLeftIndex to IdProxyR;
attach index RltLeftIndex to GeomSync;
attach index RltLeftIndex to FEMToPSI;
attach index RltLeftIndex to FEMToPS;
attach index RltLeftIndex to FEMToINS;
attach index RltLeftIndex to FEMToASM;
attach index RltLeftIndex to FEMToABP;
attach index RltLeftIndex to CADOccR;
attach index RltLeftIndex to CADtoMNR;
attach index RltLeftIndex to CADCntxt;
attach index RltLeftIndex to CADACRST;
```

2. Back up the **tmti.prd** file. Change the directory path to *MTI_ROOT* and rename the **tmti.prd** file.

   Windows example:

```
ren tmti.prd tmti.old
```

3. Change the directory path to *MTI_ROOT***meta**. Generate a new model file using the following command:

```
model -f mti.prd -b tmti.prd
```

   Copy the **tmti.prd** file to *MTI_ROOT* using the following command:

```
copy tmti.prd %MTI_ROOT%
```

4. Update the database schema as follows:

```
updatedb -o tmti.old -n tmti.prd -p pwf-file-with-full-path-name
```

Using transfer option sets to optimize performance of Data Exchange

Use transfer option sets with the **Include Modified Objects Only** option set to **true**. This optimizes the performance significantly if the data was already exported.

Configuring closure rules to optimize performance of Data Exchange

A factor is defined using a closure rule mainly containing two items and the relation between them. Each closure rule that is provided in the standard installation defines a factor. To optimize the performance of Data Exchange, each of these closure rules needs to be configured.

Following is a sample closure rule.

Field	Value
Closure Rule Clause Name	PartDoc_1
Transfer Option	Include Documents
Class1	Assembly
Class2	CADDoc
Relationship Name	DocumentsDescribingPart
Foreign Key on Class 1	
Foreign Key on Class 2	
Factor Group Number	PartDoc_1
Primary Object	Class1
Relation Type	NonSpannable
Applicable for Stubs	
Skip Secondary Objects Validation	True

In this closure rule, setting the value of the **Skip Secondary Objects Validation** field to **True** skips the validation of the secondary objects until a primary object is modified in a factor, thereby optimizing the overall performance.

Also, for optimizing the performance using this closure rule, the user must set the value of the **Include Modified Objects Only** field of a transfer option set to **True**.

14. Optional Teamcenter configuration

Using the Teamcenter Integration Framework configuration wizard

If you want to change or add sites to your Data Exchange configuration, change the mapping between sites, or specify security privileges, you can use the Teamcenter Integration Framework wizard. You access the wizard through the Teamcenter Integration Framework user interface by choosing **Configure→Sites and Monitoring** and then choose **Configure→Wizard**. In addition to the wizard, you can configure security, FMS. and Teamcenter Integration Framework properties, and monitor transactions (activity status) in the configuration interface, You can also use the **Configuration Objects** pane to manually edit the datastore configuration files.

You access the Teamcenter Integration Framework user interface in a browser at the following URL:

`http://`*`TcIF-host-name`*`:8040/controller/index`

The Teamcenter Integration Framework configuration interface provides embedded help for many of the panes.

Configuring export to secret teams

When exporting secret team objects from Teamcenter, the exporting user must have the required **Secret**, **super-secret**, or **top-secret** intellectual property (IP) clearance and a **DBA** role must exist in the **Secret** group that contains the **DBA** user. The **DBA** user must also have the required **Secret**, **super-secret**, or **top-secret** IP clearance. If the export user is not set up properly, the export fails.

Also, you must configure authorized data access (ADA) and an access control list (ACL) for the exporting users and the **IP Admin** role user. The following are requirements for exporting secret items:

- All exporting users that export secret items and the **IP Admin** role user must be in the same ACL.
- The **IP Admin** role user must be granted **IP Admin** privileges in the ACL.
- Licenses can be attached or detached to items or any valid objects only when the owning user has **IP Admin** privileges.
- Synchronizing license association between systems is not supported. Therefore, if a license is detached from an object at the owning site, the license does not get detached from the replica object.
- A license must exist at each site with the same name and the objects must be IP classified (**secret**, **super-secret**, or **top-secret**) for the license to be effective on objects at the various sites.
- A user must have **IP Admin** privileges to set the IP classification on an object.

Configuring products for editable JT files

Configuring Teamcenter Integration for I-deas

The following must be set in your Teamcenter Integration for I-deas environment:

- **IDEAS_WRITE_V9_JT** (this must be set to 1 if using V9, or 0 if using an earlier JT version)
- **includeXT = true** (this is normally set in the **tessMS.config** file)
- **Exp.XtBRep: 1** (this is normally set in the **export_param_idip.ipf** file)

For information about the Teamcenter Integration for I-deas requirements for working with modeling data and JT files, see the online help for Teamcenter Integration for NX I-deas.

Configuring Teamcenter Integration for NX

In Teamcenter Integration for NX, the following **Customer Defaults** must be set under **Teamcenter Integration for NX→General**:

- (**Assembly** tab) Choose **Structure Update on Load→Complete**.
- (**Parts without NX Datasets** tab) Choose **Add NX Datasets→On Save**.

The following defaults must be set under **Gateway→JT Files**:

- (**Import/Export** tab) Select the **Import Assembly Structure from JT File** check box.
- (**Import/Export** tab) Select the **Import Wireframe from JT Files** check box.
- (**Import/Export** tab) Select the **Import Product and Manufacturing Information from JT Files** check box.

Under **Assemblies→General** (**Miscellaneous** tab) select **Update Structure on Expand** in the **When Generating Missing Part Family Members** section.

The following options must be set in the **Assembly Load Options** dialog box:

- For **Part Versions**, select **From Folder**.
- For **Scope**, select **All Components.** and select the **Use Partial Loading** check box.
- For **Load Behavior**, select the **Generate Missing Part Family Members** check box.
- For **Reference Sets**, select **Use Model**.

- Type **Use Lightweight** in the **Add Reference Set** box.

For information about setting customer defaults and assembly load options, see the *Teamcenter Integration for NX User Guide*.

Configuring Teamcenter for editable JT files

Set the follow preferences as follows:

IMAN_UG_Foreign_Datasets

```
DatasetType="DirectModel" NamedReference="JTPART" NamedReferenceFormat="BINARY"
NamedReferenceTemplate="jt"
DatasetType="SE Part" NamedReference="SE-part" NamedReferenceFormat="BINARY"
NamedReferenceTemplate="par"
DatasetType="catia" NamedReference="catia" NamedReferenceFormat="BINARY"
NamedReferenceTemplate="model"
DatasetType="IdeasPart" NamedReference="IdeasPart" NamedReferenceFormat="BINARY"
NamedReferenceTemplate="prt"
DatasetType="CATPart" NamedReference="catpart" NamedReferenceFormat="BINARY"
NamedReferenceTemplate="CATPart"
```

NX→Show Open in NX command

Select the **Yes** check box.

15. Using Data Exchange from Teamcenter

Teamcenter Data Exchange functions

Teamcenter Data Exchange uses some menu commands that Multi-Site Collaboration uses. The configuration and destination determine whether Data Exchange or Multi-Site Collaboration performs the action.

Revising and importing exported data

If you revise data that has been exported to your site from Teamcenter Enterprise that may be returned to Teamcenter Enterprise, do not accept the prepopulated revision value. Teamcenter Enterprise does not allow a revision attribute with a decimal value, such as **A.2**. This causes the import to fail on the return to Teamcenter Enterprise. Siemens Digital Industries Software recommends that you edit the revision to an alpha value, such as **B**, to avoid import failures.

Inferring deletions

Control the infer delete function in Teamcenter using the **TIE_itemrev_infer_delete_skip** preference.

Transferring objects with attached ITAR licenses

To transfer objects with attached ITAR licenses between Teamcenter and Teamcenter Enterprise, first, ensure the license exists at the importing site. Then, use the Diverse Schema Utility to create map rules to map the license objects.

ODS operation support

Object Directory Service (ODS) operations are not supported between Teamcenter and Teamcenter Enterprise sites. All Data Exchange operations related to ODS are supported only for Teamcenter-to-Teamcenter operations. This includes publishing objects to the ODS, remote search operations, and remote import operations based on remote search results.

Export from Teamcenter

You can export local data to a remote site.

1. In the rich client, select the object that you want to export.

2. Choose **Tools→Export→To Remote using Global Multi-Site**.

3. In the **Remote To Remote Site using Global Multi-Site** dialog box, select a destination site from the **Site Selection** list.

(Optional) Type a description in the **Reason** box. The reason text is stored for future reference.

Select the desired check boxes.

Check box	Description
Immediate	Causes the transfer request to process immediately. By default, Data Exchange schedules transfer request processing based on system load conditions. Your administrator can disable this option.
Progress Indicator	When the process starts, opens the default Web browser and displays the **Audit - Activity Status** table. The table shows the activity steps as they are processed.
E-mail Notification	Requests that Data Exchange send an e-mail when a transfer is complete to the address associated with the requesting user.

4. In the **Remote Export Options** dialog box, select the desired check boxes.

> Note:
>
> The options available in this dialog box depend on the variables in the selected option set.
>
> If you are exporting a **Requirements** object, you must select the **Include All Files** check box in the **Dataset options**.

Item options	Description
Include All Revisions	Exports all revisions. When transferring site ownership, this is the only option available.
Latest Revision Only	Exports the latest revision regardless of whether it is a working or released revision.
Latest Working Revision Only	Exports only the latest working (such as nonreleased) revision.
Latest Working/Any Release Status	Exports the latest working revision, if any; if no working revision, the latest released revision with any release status is exported.
Latest Any Release Status	Exports the latest released revision with any release status.
Selected Revision(s) Only	Exports only the revisions selected in the workspace.

Item options	Description
	Caution: If you use this option to export an assembly it must contain precise parts only. If you attempt to export an assembly that contains imprecise parts, alternates, or substitutes, the export fails.
Specific Release Status Only	Exports only the latest revision with the specified release status selected from the list.

Product structure options	Description
Include Entire BOM	Includes all components if the item selected is an assembly. The revision selectors allow you choose which revision to export with the selected item and its component items, if applicable. You can choose only one revision selector. The **TC_bom_level_export** preference controls whether this option is available.
Transfer Top-Level Item Only	Transfers the selected assembly item and export all components with no site ownership transfer.

Dataset options	Description
Include All Versions	Includes all dataset versions with each dataset selected for import or export. When this option is not set, includes only the latest version of each dataset selected for import or export.
Include All Files	Includes all underlying operating system files (such as named references) with each dataset selected for import or export. If you do not set this option, only the dataset metadata is imported or exported.

Save options	Description
Save All Option As Default	Saves the options you select in the dialog box as the default option settings.
Restore Option Set Defaults	Replaces all options settings with the defaults settings.

Session options	Description
Include Modified Objects Only	Clear (not checked).

Session options	Description
	If selected, only objects that have been modified since the last transfer are exported. Caution: If you select the **Include Modified Objects Only** check box, you must not select **Transfer Top Assembly** as the BOM option. If data in Teamcenter is replicated to a remote site: • You cannot delete the replicated objects and synchronize them to remote sites using the **Include Modified Objects Only** option. You must export the full object set to delete the replicated objects. • You cannot synchronize **license_id** objects or structure effectivity using the **Include Modified Objects Only** option. You must export the full object set to update this information.
Transfer Ownership	Transfers site ownership to the target site. When the this option is not selected, your site retains ownership. If you transfer an item revision with a sequence, its sequence manager is also transferred. The **TC_ownership_export** preference controls the default value of this option. Siemens Digital Industries Software recommends that you leave the default setting for this option to unselected.
Continue On Error	Allows the remote import or export objects operation to continue if errors are encountered while importing/exporting optional objects. All objects are considered optional except the following: • **Requirements** • **Specifications** • **Item Master** • **Item Revision Master** This option is disabled if the **Transfer Ownership** option is set.

5. Click **OK** to close the dialog box and click **Yes**.

Check out to a remote site

You can check out an object to a remote site so that it can only be modified by the remote site after it is imported through a briefcase package. This creates a reservation object and associates it with the checked-out object. You cannot check out an object to a site if it is already checked out either locally or to another site. If you Log on to a hub site, you can also check out a replica object to remote site.

1. In My Teamcenter, select one or more objects that you want to check out to a remote site.
2. Choose **Tools→Site Check-In/Out→Check-Out to site**.
3. In the **CheckOut to site** dialog box, type an identifier in the **Change ID** box and select the desired remote site from the **Target Site** list.
4. (Optional) Type information in the **Comments** box.
5. (Optional) Click  to display the **Explore** dialog box. Use the navigation tree to add selection for any related objects that you want checked out to the site.
6. Click **OK**.

Cancel checkout to a remote site

If you checked out an object out to a remote site (see *Check out to a remote site*), you can cancel the checkout by choosing **Tools→Site Check-In/Out→Cancel Check-Out to site**. You can also cancel checkout of a replica object. Only an administrator can cancel a site checkout when another user performed the checkout. When a site checkout is canceled:

- If it was checked out to another site, the reservation object is restored to its pre-checkout state.
- If the object was not checked out to another site, the reservation object is removed.
- The export record is removed.
- Datasets are not restored their pre-checkout state. The only action is the site checkout lock is removed.

Check in from a remote site

You can check in an object to your site if it is checked out to your site from a remote site (see *Check out to a remote site*). You can also check in a replica object. When an object is checked in to a site:

- If it was checked out to another site, the reservation object is restored to its pre-checkout state.
- If the object was not checked out to another site, the reservation object is removed.

- The export record is updated.

1. In My Teamcenter, select the objects that you want to check in to your site.
2. Choose **Tools→Site Check-In/Out→Check-In from site**.
3. Click **Yes**.

Check replica synchronization

The replica sync state of an object identifies whether it is up to date with the primary object.

1. Select the replica objects that you want to check.
2. Choose **Tools→Multi-Site Collaboration→Check Replica Sync State**.

   Teamcenter displays the **Replica Sync State** dialog box showing replicated object status.

16. Using Data Exchange from Teamcenter Enterprise

Data exchange capabilities and prerequisites

Before you export any objects from Teamcenter Enterprise,

You use the Data Exchange solution to transfer data from Teamcenter Enterprise to Teamcenter. Before exporting the data, you must generate the XML schema file for the mapper and create your mappings or generate the Map Control Files (MCFs) using this XML schema file. Additionally, you must create sites, transfer modes, transfer option sets, and closure rule clauses.

Also ensure that administrative tasks such as creating users, groups, projects, ADA licenses, and so on in Teamcenter are completed.

You can export data from Teamcenter Enterprise to Teamcenter either with ownership or for reference. If you are exporting the data for reference, the ownership of the data remains with Teamcenter Enterprise and only a replica is sent to Teamcenter. You can also swarm the data to be exported. Swarming is the process of splitting a larger transaction into smaller subtransactions and executing them simultaneously.

During a reference transfer of data, if you want that the ownership of the data to be transferred later, you select a transfer option set that has a switch. To transfer the ownership of the data, the administration uses the **Flip The Switch** action.

> Caution:
>
> A configuration item in Teamcenter Enterprise is an administration object. Therefore, you must ensure that you do not have configuration items and business items in Teamcenter Enterprise with the same ID in any data that may be exported. This situation can cause a transfer to fail because the item already exists at the Teamcenter site.

XML schema file

As an administrator, you must generate an XML schema file before exporting an object. This is because the mapper needs this XML schema file to map to the schema of the remote installation.

Every time you add or modify classes in Teamcenter Enterprise, you must generate a new XML schema. You can specify the directory where the schema files must be generated in the configuration file.

Before you generate the schema, you modify the following attributes of the *GMS configuration object:*

Attribute name	Description
Directory Path For Schema File	Specifies the directory where the generated schema files will be stored. Select one of your work locations or any managed locations.
Schema File Name	Specifies the name to be given to the schema file when it is generated.

To generate the XML schema file in classic client, select **Action→Administrative Actions→GMS Actions→Generate XML Schema**.

To generate the XML schema file in thin client, click **System Utilities→Generate XML Schema** in the navigation bar in the Administration Editor.

The generated XML schema file includes another XML schema file that is available out of the box. The name of this schema file is **TCEntBaseSchema.xsd** and it is located in the *MTI_ROOT***/xml/ems** directory. You must copy this file and paste in the same directory where you generated the XML schema file.

Exporting data from Teamcenter Enterprise to Teamcenter

Exporting data for ownership transfer

When you perform an ownership transfer from Teamcenter Enterprise to Teamcenter, the ownership of the data is transferred to Teamcenter and only a replica remains in Teamcenter Enterprise. Therefore, you cannot modify the data in Teamcenter Enterprise. To perform an ownership transfer, you must select the **Transfer Ownership** check box during export.

If you want to transfer the ownership of the data later, first perform a reference transfer and then transfer the ownership using the flip-the-switch feature.

> Note:
>
> Ownership transfer is not supported for the dry run for export to a remote site function.

Exporting data for reference

When you export data from Teamcenter Enterprise to Teamcenter for reference, the ownership of the data remains with Teamcenter Enterprise and only a replica is transferred to Teamcenter. If you want to transfer the ownership of the data later using the flip-the-switch feature, you must use the switch object.

Caution:

For CATIA (**CATDrawing**) drawings, Teamcenter compares the drawing name to the assembly or component part number during the import process. Teamcenter uses this comparison to determine whether to associate the drawing with the item revision (**ItemRevision**) of the assembly or component or to associated it with its own **Item**/**ItemRevision** object.

If an exported component or assembly contains more than one related drawing file, and one of them matches the assembly or component part number, new revisions may not be replicated properly when they are imported into Teamcenter.

To ensure that revision are attached to the correct **Item**/**ItemRevision** object, all related drawing files must have different names from the related assembly or component.

Configuring the root object for export

Valid root objects for export

Validity of an object as a root object for export depends on the class constant **SelectedAsRootObjectC** for the corresponding class. By default, this class constant is set to **FALSE** for the **PdmRoot** class and is set to **TRUE** for the **PdmItemEms** class. Therefore, objects of all sub classes of the **PdmItemEms** class are valid as root objects for export except for the **ProData** class and its sub classes.

When you export an object from Teamcenter Enterprise, the export action submits the export request to Teamcenter Integration Framework. Teamcenter Integration Framework schedules the transfer and starts the Teamcenter Enterprise exporter. Before submitting the export request, the following validations are performed:

- The selected objects are validated against the message access rules.
- The site is validated.
- The compatibility of the selected options is validated.
- The compatibility of the selected objects with the selected options is validated.

For additional customization, the **ValidateExportDialog** published message can be used.

Once started by Teamcenter Integration Framework, the Teamcenter Enterprise exporter processes the export by:

1. Preparing the objects for export.
2. Gathering the objects for export.
3. Processing the objects for export.

4. Completing the export action.

Preparing objects for export

When the export process begins, the root objects are validated against the message access rules (MARs) for the exporting user. The objects for which the user does not have permissions are ignored. The export process continues if the user has permission to export at least one object.

> Note:
>
> For the Single-Sign-On (SSO) mode, the exporting user is the same as the user who submitted the export request to Teamcenter Integration Framework from Teamcenter Enterprise.
>
> For the non-SSO mode, the exporting user is the user that is configured in Teamcenter Integration Framework.

The valid root objects then gather additional root objects using the **GetMultipleRootObjects** published message.

Following are the default implemented behaviors of the **GetMultipleRootObjects** published message for different classes:

> Note:
>
> The default behavior implementation is based on the version option of the transfer formula.
>
> The **GetMultipleRootObjects** published message can be used to customize the default behaviors.

- **PdmItemEms** — Copies the input object to the output set.
- **StrucBl** — For the **Selected Version** option, copies the input object to the output set.
  For the **Latest Version** and **Include All Versions** options, invokes the same API on the primary object.
- **ItemMstr** — For the **Latest Version** option, returns the latest revision of the object.
  For the **Include All Versions** option, returns all revisions of the object.
  For the **Selected Version** option, returns error (mfail).
- **GenDoc** — For the **Selected Version** option, returns the input object.
  For the **Latest Version** option, returns the latest revision of the document.
  or the **Include All Versions** option, returns all revisions of the document.
- **Part** (for ProE) — Returns the **ProData** attached to the latest document related to the part.

Some other tasks that are performed to prepare the objects for export are initializing the transaction log, validating the target site, loading appropriate closure rules for further usage, and so on.

Gathering objects for export

The root objects prepared for export are processed for island-wise object gathering by applying appropriate closure rule clauses to the root objects as well as to the objects gathered thereafter. Once the gathering for island process is complete, the objects of the island are transferred to a store that holds the objects either in a cache or in the Exporter Spill Database based on the configuration (**Exporter Cache Size**). While transferring the island to the store, the objects of the island are merged with any existing island of the store, if required.

For object gathering, the closure rule clauses related to the transfer mode of the transfer option set (TOS) are used. Out of these, closure rule clauses that have **Class1** same as the object's class are applied on the object to gather additional objects.

Closure rule clauses are applied in the following way:

- Closure rule clauses having no transfer option (also known as default closure rule clause) are applied to gather additional objects.
- Closure rule clauses having transfer option as per options in TOS are applied to gather additional objects.
- Rest of the closure rule clauses are applied to gather stubs (also known as Not-In-Formula objects). For Non-StrucBIs, there is a special case of Not-In-Formula object gathering based on **GsSubLabel**. The objects of the series that were transferred to the same site earlier with same **GsSubLabel** populated are gathered as Not-In-Formula objects for the Non-StrucBI objects.

While expanding any closure rule clause, if it has the **Filter Class2 object** attribute selected, the gathered objects and relations are sent to the **FilterOtherSideObjAndRel** published message for filtering. For modifying the default behavior, this published message can be implemented appropriately.

For the objects for sub-classes of the **WrkImEms** class, other than closure rule expansion, there is a special gathering in specific cases. If the object or the latest object of the series has unconfirmed ownership export for the target site then the latest object is added to the island of the object. However, the version option should not be **Selected Version**. The objects of the series that are already transferred or have pending transfers as full object to the target site, are gathered by invoking the **GetObjectsInSeriesForOwnTransfer** published message on the latest object. The **GetObjectsInSeriesForOwnTransfer** published message internally uses the **GetPredForOwnershipTransfer** published message.

During the gathering phase, apart from object gathering, stubs are identified. Stubs are the objects that cannot be exported as full objects because of reasons like being owned by some other site, export protected, message access rule protected, and so on.

If the TOS has the **Include Modified Objects Only** option selected, a factor is not exported if all the objects in the factor are not modified after the last export. Else, all the objects in the factor are exported. If the **SkipSecObjects** attribute is selected in the closure rule clause, the secondary objects are not considered while deciding whether to export the factor.

Note:

If multiple closure rule clauses are grouped as a single factor then all the closure rule clauses of the factor must have the same value for the **SkipSecObjects** attribute.

When the factor is exported, the following sequences summarize the fullness of the objects:

- Relations are exported as full objects.
- Objects that are owned by different site, export protected, or message access rule protected are exported as stubs.
- Primary object of the factor is exported as full object.
- Objects that are gathered by not-in-formula expansion are exported as stubs.
- Objects that are modified after the last export are exported as full objects.
- Objects that are not modified after the last export are exported as stubs.

An object is considered modified if:

- The object was never exported.
- The object was earlier exported as stub.
- The earlier export is not confirmed yet.
- The export record is marked as dirty. Objects having non-stub confirmed export records with the **IsDirty** attribute set to **FALSE** are considered not modified.

  The export record is marked as dirty in the following cases:

  - If you update an object to be exported for reference. In this case, the **IsDirty** attribute is set to **TRUE**.
  - In case of creating, deleting, or updating any relation, if the left object is exported for reference, then the export record of the left object is marked as dirty.

    If you create, delete, or update any relation. In this case, if the left object is exported for reference, the export record of the left object is marked as dirty.

> Note:
>
> Marking of export records as dirty is performed by the EMS event server, so ensure that the EMS event server is not stopped.

Another task performed during the gathering phase is swarming. Based on the swarming configurations, if required, further island gathering is stopped, and requests for sub-transactions are submitted to Teamcenter Integration Framework.

Important messages used during the object gathering phase

DoExport is the first level message that is invoked by Teamcenter Integration Framework on the **MultisiteServiceHandler** class.

ExecuteExportProcess, responsible for data gathering, gathers the objects to be exported by invoking the recursive message, **GatherIslands**.

Starting with a root object, the **GatherIslands** message invokes the **PopulateIsland** message to gather objects for the island. After gathering objects for the island, the **GatherIslands** message performs the island merge by invoking the **MergeIsland** message. Later, it transfers the objects of the island to the store by invoking the **WriteToTHS** message.

The following example shows the message call hierarchy:

```
1 DoExport @ MultisiteServiceHandler (MultisiteServiceHandler:DoExport)
  2 ExportObjects2 @ Exporter (Exporter:ExportObjects2)
    3 ExportObjectsInt @ Exporter (Exporter:ExportObjectsInt)
      4 ExecuteExportProcess @ Exporter (Exporter:ExecuteExportProcess)
        5 GatherIslands @ Exporter (EMSStoreProcessor:GatherIslands)
          6 PopulateIsland @ Island (Island:PopulateIsland)
          6 MergeIsland @ Island (Island:MergeIsland)
          6 WriteToTHS @ Island (Island:WriteToTHS)
          6 GatherIslands @ Exporter (EMSStoreProcessor:GatherIslands)
```

The **PopulateIsland** message not only gathers the object for a specific island but also gathers the entry points to other islands (also referred to as ports of other islands). After successfully transferring the island objects to the store, the **GatherIslands** message invokes itself for each of the ports gathered for further island gathering.

The **PopulateIsland** message gathers objects by invoking the following messages:

- **GatherInsideIsland** message that expands the non-spannable closure rules.
- **GatherOutsideIsland** message that expands the spannable closure rules.

- **AddObjsOfPreviousTransfersToIsland** message that add the required objects from the previous transfers.

The following example shows the messages invoked by the **PopulateIsland** message:

```
6 PopulateIsland @ Island (Island:PopulateIsland)
  7 GatherInsideIsland @ Island (Island:GatherInsideIsland)
  7 GatherOutsideIsland @ Island (Island:GatherOutsideIsland)
  7 AddObjsOfPreviousTransfersToIsland @ Island (Island:AddObjsOfPrevious
TransfersToIsland)
```

The **GatherInsideIsland** message invokes the **GatherNonSpannableFactors** message on each object of the island. After identifying the applicable closure rules for an object, the **GetObjects** message is invoked on the object for each closure rule. The **GetObjects** message expands one closure rule on a given object to gather additional objects.

The following call sequences are responsible for actual closure rule expansion:

```
7 GatherInsideIsland @ Island (Island:GatherInsideIsland)
  8 GatherNonSpannableFactors @ Island (Island:GatherNonSpannableFactors)
    9 GetScopedObjects @ Scoper (Scoper:GetScopedObjects)
      10 GetObjectsForClosureRules @ Scoper
(Scoper:GetObjectsForClosureRules)
        11 GetObjects @ Scoper (Scoper:GetObjects)
```

Once the object gathering phase is complete, objects of each island are processed by the **ProcessIslandFromStore** message that decides the transfer intent of the island, create or updates the records, and writes the objects into the export XML file.

The following is a sample call sequence:

```
3 ExportObjectsInt @ Exporter (Exporter:ExportObjectsInt)
  4 WriteExportContent @ Exporter (Exporter:WriteExportContent)
    5 ProcessObjectsFromStore @ ExpSpill
(ExpSpill:ProcessObjectsFromStore)
      6 ProcessIslandFromStore @ ExpSpill
(ExpSpill:ProcessIslandFromStore)
        7 ProcessIslandForExport @ Island (Island:ProcessIslandForExport)
          8 DecideIslandIntent @ Island (Island:DecideIslandIntent)
          8 ProcessRecordsForExport @ Island
(Island:ProcessRecordsForExport)
          8 WriteIslandToXML @ Island (Island:WriteIslandToXML)
```

Processing objects for export

Objects gathered are processed island by island for intent evaluation, record keeping, and writing to export XML.

Once the object gathering phase is complete, objects of each island are processed by the **ProcessIslandFromStore** message that decides the transfer intent of the island, creates or updates the records, and writes the objects into the export XML file.

The following is a sample call sequence:

```
3 ExportObjectsInt @ Exporter (Exporter:ExportObjectsInt)
  4 WriteExportContent @ Exporter (Exporter:WriteExportContent)
    5 ProcessObjectsFromStore @ ExpSpill
(ExpSpill:ProcessObjectsFromStore)
      6 ProcessIslandFromStore @ ExpSpill
(ExpSpill:ProcessIslandFromStore)
        7 ProcessIslandForExport @ Island (Island:ProcessIslandForExport)
          8 DecideIslandIntent @ Island (Island:DecideIslandIntent)
          8 ProcessRecordsForExport @ Island
(Island:ProcessRecordsForExport)
          8 WriteIslandToXML @ Island (Island:WriteIslandToXML)
```

For all objects being exported, export records are either created or updated for the target site. These records are marked as unconfirmed and are later confirmed during the confirm export step.

Intent evaluation

The intent of the island is decided based on user input during the export request submission.

If the user intends to transfer ownership, it is validated that the ownership for all the objects in the island can be transferred. When the island is found to be invalid for ownership transfer, if the **Continue On Error** option is selected, the island is evaluated for reference intent. If the **Continue on Error** option is not selected, the export action is aborted.

By default, the island is considered invalid for ownership transfer in the following situations:

- If the latest revision of any object is not being exported and is not already owned by the target site.
- If any object is reserved, is owned by a different site, or has unconfirmed ownership to a different site.
- If any object is immutable because of some action outside of Teamcenter Enterprise Data Exchange.

> Note:
>
> Relations and stub objects of an island are not considered while deciding the intent of the island.

If the user intention is to transfer for reference, the island is evaluated for reference intent. It may so happen that it becomes invalid for reference.

Example:

There are multiple revisions of an object and the latest revision is transferred for ownership. If the user now selects some non-latest revision and transfers that revision with selected version transfer option set, the island of the object will be considered invalid for reference transfer.

In such situations, irrespective of the continue on error option, the island is evaluated for ownership transfer and if valid, the ownership of the island is transferred. This is known as the **Bump up** case where the island intent is automatically bumped up to ownership transfer from reference transfer.

Writing to export XML

All objects being exported are written into the export XML. Different configurations and custom hooks are available to control the attributes that get exported for any object.

- Attributes that have **noexport** tag defined are not exported.
- Attributes added to the **IgnoreAttributeList** value set are not exported.
- System attributes are either exported or skipped based on the value of the **Export System Attributes** configuration variable of the GMS configuration object.
- By default, only a selected number of dynamic attributes are exported. To export any additional dynamic attributes, see *Configuring class constant for exporting dynamic attributes of an object*.

  Dynamic attributes exported by default

Class	Dynamic attributes
WorkItem	**DestinationFolderName**
StrucBl	**ViewNetwork**, **StartingViewName**, **ViewInViewNetwork**, **EligibleViewSet** (these are in addition to what is available in **WorkItem**)
VewDepR	**CurrentViewNetwork**, **EligibleViewSet**
OccCfgR	**RelMTID** (this is in addition to what is available in **VewDepR**)

- When an object is exported as stub, the stub must have certain attributes so that it can be meaningfully used in data mapping. The **StubClassC** class constant can be used to assign a template class representing the stub attributes to a specific class. For example, you can refer to the **stbems.prd** file under *MTI_ROOT***/meta/ems** folder.

  In addition to these configurations, the **ExportObjectToXMLPre**, **ExportObjectToXMLPost**, **WriteXMLRootTagPost**, **WriteXMLEndTagPre**, **WriteXMLAttrsPost**, and **WriteXMLSubElements** published messages are available for customizing the export XML at different stages of writing the object into the export XML.

Completing the export action

To complete the export action, the following tasks are performed:

- Closing the transaction log.
- Generating the FMS ticket for export XML and transaction log.
- Preparing the result object and pass it to Teamcenter Integration Framework for data mapping.

Export data from the classic client

1. Select the objects you want to export.

   Note:

   For Classic Client ProE Integration (CCPE), you can export data items that belong to the following classes only:

   - **ProLay**
   - **ProSec**
   - **ProFrm**
   - **ProDgm**

2. Choose **Actions→GMS Actions→Export to Remote Sites**.

3. In the **Remote Export Options** dialog box, enter the required information. The following table lists and describes the fields in the **Remote Export Options** dialog box.

Fields	Description
Site Name	Specifies the name of the site where the object is to be exported.
Transfer Option Set	Specifies the transfer option set name.
Reason for Transfer	Specifies a justification for the transfer.
Immediate	Requests Teamcenter Integration Framework to export the object immediately.
Progress Indicator	Enables progress monitoring for the request.
E-mail notification	Sends an e-mail to you upon completion of the export.
Transfer Ownership	Transfers ownership of the object to the remote site.

Fields	Description
	Note: If you select the **Transfer Ownership** check box while exporting data, the switch name attached to the selected TOS is ignored for that transaction.
Continue on Error	Allows the export transaction to continue if recoverable or nonfatal errors are encountered while exporting objects. If you want to use the flip-the-switch feature later to transfer the ownership of the objects, you must select this field while performing a reference transfer. Also, to enable swarming, you must select this field.

4. Click **OK**.

Export data from the thin client

1. Select the check boxes next to the objects you want to export.

2. Choose **More Actions→Export to Remote Sites**.

   The **Remote Export Options** pane displays two set of objects. One set lists the objects you selected that are valid for export. The other displays the objects you selected that cannot be exported. From the set of the valid objects, you can deselect the objects that you do not want to export.

3. In the **Remote Export Options** pane, enter the required information. The table in the *Export data from the classic client* section describes the fields in the **Remote Export Options** pane.

4. Click **OK**.

Export data from Team Browser

1. Choose **Tools→Search** to search for the objects you want to export. On the search results pane, select the objects and click **Remote Actions→Export to Remote Sites**.

2. In the **Remote Export Options** dialog box, enter the name of the site to which you want to export the data in the **Site Name** box.

3. In the **Transfer Option Set** box, enter the name of the transfer option set.

4. Select the relevant check boxes. The table in the *Export data from the classic client* section describes these check boxes.

> Note:
>
> When you select an IDS data model item, the item must be a component or an assembly. Data Exchange does not support the export of any other type as a root item.

5. Click **OK**.

Swarming data for export

Swarming is implemented when the value of the **Ideal Dispersion Rate** value at the remote site is greater than 0 (zero). Also, to enable swarming, the **Continue on Error** value must be set to **true** while exporting data. The approximate number of objects to be exported per transfer is decided by the **Preferred Transfer Size** value at the remote site. These values are set in Teamcenter Enterprise when you create the site object.

> Note:
>
> Swarming is not supported for the dry run for export to a remote site function.

While processing an export request, the exporter stops gathering islands after the size of the gathered objects reaches the preferred transfer size. If the preferred transfer size is reached in the midst of gathering an island, the exporter gathers all the objects in that island and then stops gathering the remaining islands.

The exporter sends export subrequests for the remaining ports of the island that are not yet processed. The number of export subrequests to be initiated is the same as the ideal dispersion rate.

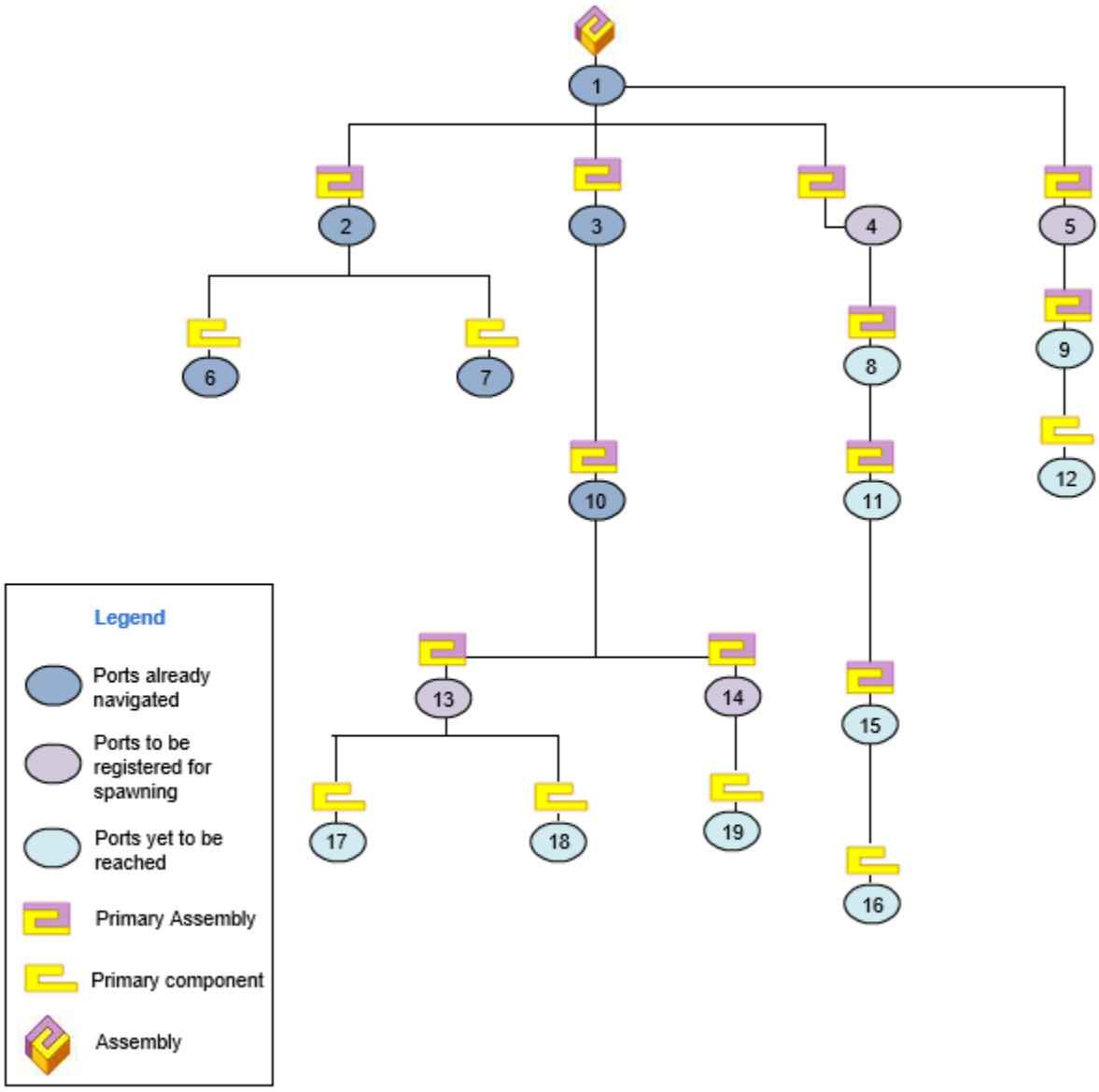
For example, if there are 10 ports to be swarmed and the ideal dispersion rate is 2, the exporter initiates 2 concurrent export subrequests each having 5 objects. While submitting the export subrequests, the exporter also sends an identifier. The value of this identifier is the same for the initial request and for all associated export subrequests.

Simultaneously, the gathered islands are written in an **XML** file and sent to Teamcenter Integration Framework for mapping.

> Note:
>
> An export subrequest can further send subrequests if the number of objects gathered exceeds the preferred transfer size.

Swarming a structure

This section discusses the way a sample structure is swarmed when the value of the preferred transfer size is constant and the value of the ideal dispersion rate varies.



The sample structure represents three objects namely item master (primary), revision, and item revision relation. Each connector between two ports represents a relation object. Consider that the preferred transfer size is 26. While processing an export request, the exporter starts gathering objects starting from port 1. When the exporter reaches port 10, the total number of objects gathered are 27. The following equation shows how the objects gathered equals to 27:

6 ports * 3 objects per port + 9 relations = 27

As the preferred transfer size is 26 and the number of objects gathered is 27, the exporter completes the current export request by writing the navigated contents in an **XML** file. Simultaneously, the exporter groups the remaining ports that are yet to be navigated. Based on the ideal dispersion rate, the exporter distributes the ports into different export subrequests.

Preferred transfer size	Ports to be swarmed	Ideal dispersion rate	Number of concurrent export subrequests swarmed	Root object sets
26	13, 14, 4, 5	1	1	One set with four objects.
		2	2	Two sets with two objects each.
		3	3	Three sets, where one set contains two objects and the remaining two sets contains one object each.
		4	4	Four sets with one object each.
		5	4	Four sets with one object each.

Teamcenter Enterprise and Teamcenter Integration Framework identify the initiated export request and the associated export subrequests using the initial identifier.

Dry run for export from Teamcenter Enterprise

Generating sample Teamcenter Enterprise XML files

As an administrator, you can perform a dry run for export before exporting the objects. The dry run for export feature generates the sample XML files from Teamcenter Enterprise. You may require these sample XML files for creating or designing the mapping. Also, while designing or configuring closure rules in Teamcenter Enterprise, you may not want to transfer the data to other systems but may want to know the affect or adding or updating closure rules. The dry run for export feature generates the sample XML files without transferring the data to the Teamcenter.

You can use this feature independently from your Data Exchange environment. An operational transition configuration with Teamcenter Integration Framework and Teamcenter is not required to generate sample export XML files.

Because a complete operational transition environment is not required, this function does not change any objects and does not create or update any export or import records. Additionally, it does not support the following transfer features that required a full setup:

- Switch actions (flip the switch functionality)
- Swarming

- Transfer of site ownership
- Include modified objects only

Perform a dry run for export from the classic client

1. In the classic client, search for an object that you want to export.
2. Select the object and choose **Action→GMS Actions→Dry Run for Export to Remote Site**.
3. Enter the required information and click **OK**.

   A single export XML containing the exported data is created in the vault location of the remote site.
4. The dry run for export to remote site action fails for objects that are not available in a vault.

   The dry run completion message is displayed in the status window.

Perform a dry run for export from the thin client

1. In the thin client, search for an object that you want to export.
2. Select the object and choose **More Actions→Dry Run for Export to Remote Site**.
3. Enter the required information and click **OK**.

   A single export XML containing the exported data is created.
4. The dry run for export to remote site action fails for objects that are not available in a vault.

   If the dry run is successful, the information about the host name, location of the export XML and transaction log file is displayed in the status window.

Transferring ownership of reference objects as a group

Using a switch during reference transfer

A switch is used for grouping objects during a reference transfer so that they can be flipped for ownership transfer later. You flip the ownership using the **Flip The Switch** action. When you perform this action, instead of exporting all the objects, only the ownership of the objects are transferred. This ensures a faster ownership transfer as compared to exporting all the objects for ownership.

> Note:
>
> The flip the switch action is not supported for the dry run for export to a remote site function.

While exporting objects as replicas, you can select:

- A transfer option set (TOS) that is attached to a switch. In this case, the objects get transferred to Teamcenter and also get attached to the switch. Later, using the **Flip The Switch** action, you can transfer the ownership of these objects to Teamcenter.

  > Note:
  >
  > If the TOS has the **Include Modified Objects Only** check box selected, all the objects get attached to the switch and only the objects that are modified get exported. The include modified objects only option is not supported for the dry run for export to a remote site function.

- A TOS that is not attached to a switch. In this case, before the you perform the **Flip The Switch** action to transfer the ownership of the objects, you must attach the objects to a switch.

The **Flip The Switch** action transfers the ownership of objects that are already transferred as replicas to the remote site. Before performing this action, you must ensure that all these objects are attached to a switch.

Before performing the **Flip The Switch** action, it is recommended that the you perform the **Dry Run For Flip The Switch** action.

You can perform the **Flip The Switch** action in classic client only.

> Note:
>
> A switch can be flipped successfully only once.
>
> You cannot delete a switch after it is flipped. Also, you cannot delete a switch if it is attached to any objects.

You can also get all the objects attached to a switch and remove objects that are attached to a switch.

Objects attached to a switch

For objects that are exported as replicas without attaching them to a switch, before the administrator flips the ownership of these objects, you must attach them to a switch using the **Attach to Switch** action. You cannot attach objects to a switch that is already flipped successfully.

You can also attach objects to a different switch, if required. Attaching an object to a new switch, detaches the object from the switch it was earlier attached to.

All objects in an island are attached to the same switch.

If you try to attach an object that is not yet exported, to a switch, an error message stating that the object was never exported is displayed.

Once you attach an object to a switch, you can remove the object from the switch by performing the **Detach From Switch** action.

You can perform the **Attach to Switch** action in classic client only.

> Note:
>
> You can attach only objects sent as full objects and not as stubs, to a switch.

Attach objects to a switch

1. In classic client, select the objects that you want to attach to a switch.
2. Choose **Action→GMS Actions→Attach to Switch**.
3. In the **Attach To Switch** dialog box:
   a. In the **Site Name** box, enter the name of the target site where you want to transfer the objects to.
   b. In the **Switch Name** box, enter the name of the switch to which you want to attach the objects.
   c. In the **BOM Option** box, specify whether you want to include the entire BOM or want to transfer the top assembly only.
4. Click **OK** to attach the objects to the switch.

Get objects in a switch

1. In classic client, search for a switch.
2. Select the required switch.
3. Choose **Action→GMS Actions→Get Objects For Switch**.
4. The **Object Browser** dialog box lists all the objects that are attached to the selected switch.

Detach objects from a switch

You can detach objects from a switch by performing the **Detach from Switch** action. This action detaches the selected object, and all objects that are part of the same island as the selected object, from the switch. You cannot detach an object from a switch that has already flipped successfully.

You can perform this action in classic client only.

1. In classic client, select the objects that you want to detach from a switch.

2. Choose **Action→GMS Actions→Detach from Switch**.

Preparing objects attached to a switch for ownership transfer

Dry run for ownership transfer

Before flipping the switch for ownership transfer, the administrator can perform a dry run process. The dry run process for flipping the switch gathers all the objects attached to the switch and validates them. A report is then generated listing objects that cannot be flipped as they may have been modified since the last export. The report contains the display name of the objects that are modified.

Based on the report, the administrator can choose:

- To go ahead with flipping the ownership of the objects. In this case, you must remove the objects that cannot be flipped from the switch.

- Not to flip the ownership of the objects till the **Flip The Switch** action is successful for all the objects in the switch. In this case, you re-export the objects that failed so that the administrator can flip the ownership later.

The administrator can perform the **Dry Run For Flip The Switch** action in classic client only.

> Note:
>
> The dry run for flip the switch process does not modify the data base and does not generate the flip XML file.

Search for a switch

1. In classic client, choose **Query→Administrative Classes→GMS Classes→Switch**

2. In the **Query for Switch** dialog box, enter the search criteria and click **OK**.

3. The **Switch Objects** dialog box lists the available switches.

Perform a dry run for ownership transfer

1. In classic client, search for a switch.
2. Select the required switch.
3. Choose **Action→Administrative Actions→GMS Actions→Dry Run For Flip The Switch**.
4. If the dry run is successful, the following message is displayed In the bottom pane of the **Switch Objects** dialog box:

   Switch-name `is ready for 'Flip the switch' action.`

Flip the ownership of objects

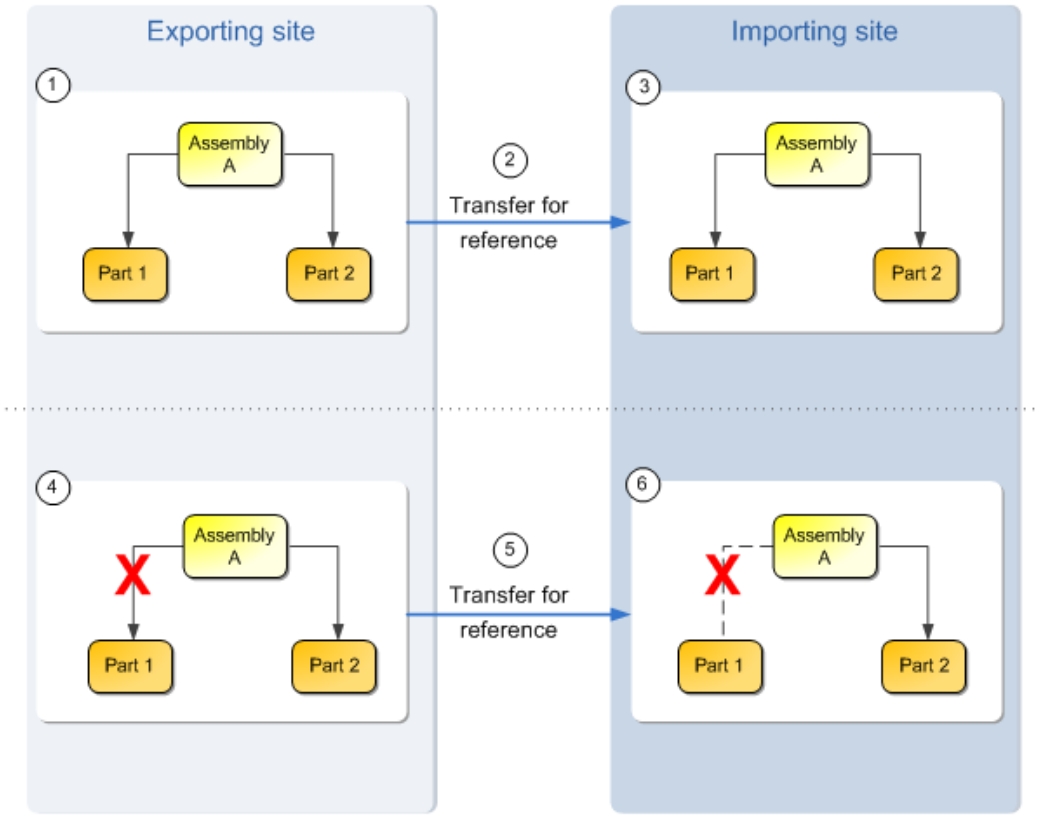
1. In classic client, search for a switch.
2. Select the required switch.
3. Choose **Action→Administrative Actions→GMS Actions→Flip The Switch**.

Teamcenter Enterprise importer

Infer delete

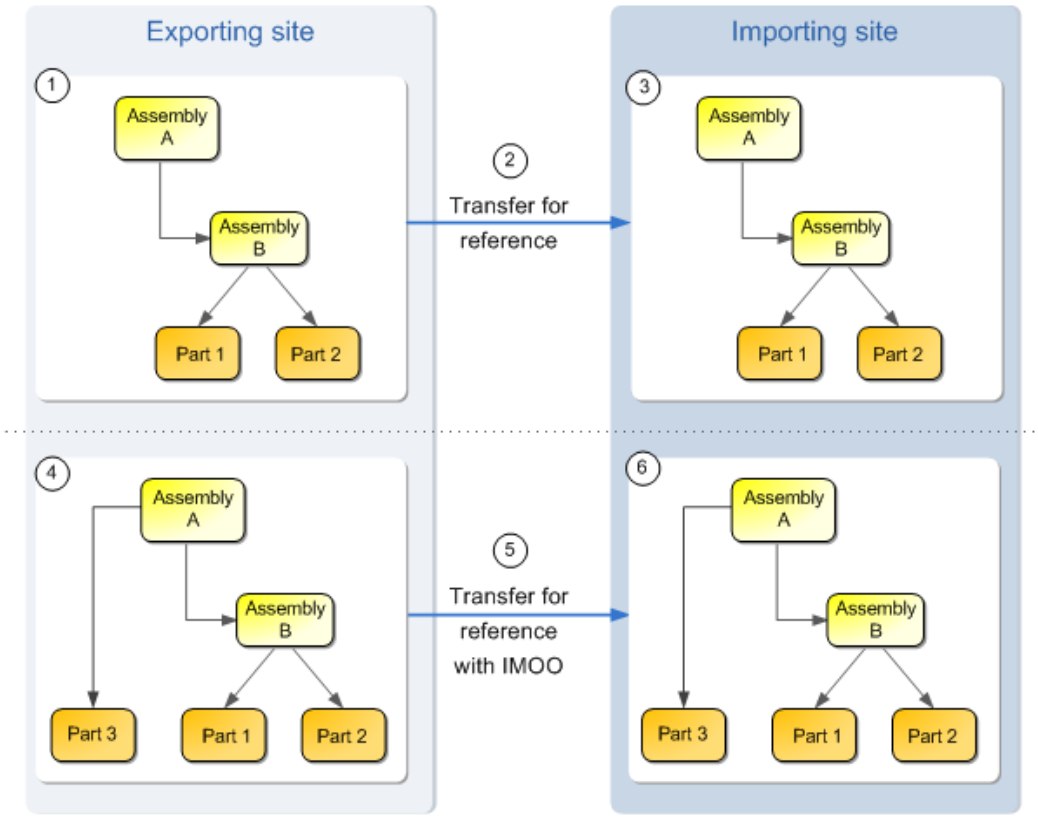
Infer delete is a mechanism of deleting relations between objects that were present in the earlier import but not present in the current import. This mechanism informs the importing site about any relation deletion that occurred in the exporting site.

The following example applies to importing TC XML data into Teamcenter:



Step 1	The exporting site has an assembly, *Assembly A*, with two parts, *Part 1* and *Part 2*.
Step 2	The exporting site transfers *Assembly A* as full object for reference.
Step 3	The importing site receives *Assembly A* along with *Part 1* and *Part 2*.
Step 4	The exporting site deletes the relation between *Assembly A* and *Part 1*.
Step 5	The exporting site transfers *Assembly A* as full object for reference.
Step 6	The importing site receives *Assembly A* with only *Part 2* related to it. As the importing site receives *Assembly A* as full object but does not receive *Part 1*, it infers that the relation between *Assembly A* and *Part 1* is deleted. Therefore, the importing site also deletes the relation between *Assembly A* and *Part 1*. However, it does not delete *Part 1* itself.

The following example applies to importing TC XML data into Teamcenter Enterprise only. When an object is sent as a stub, the importing site does not delete the relation starting from the object.



Step 1	The exporting site has an assembly, *Assembly A*, with a subassembly, *Assembly B*. *Assembly B* has two parts, *Part 1* and *Part 2*.
Step 2	The exporting site transfers *Assembly A* as full object for reference.
Step 3	The importing site receives *Assembly A*, *Assembly B*, *Part 1*, and *Part 2*.
Step 4	The exporting site adds a new part, *Part 3*, to *Assembly A*.
Step 5	The exporting site transfers *Assembly A* as for reference with the **Include Modified Objects Only** option selected. *Assembly B* is sent as a stub, and *Part 1* and *Part 2* are not exported.
Step 6	The importing site receives *Assembly A* along with its relations to *Part 3* and *Assembly B*. As *Assembly B* is received as a stub, the importing site does not infer delete its relations with *Part 1* and *Part 2*.

In Teamcenter Enterprise, infer-delete of ancestor relations is controlled by the **InferDeleteAncestorRelationC** class constant. In the default configuration, this class constant is set to **-** for the **RelationEms** class. In other words, the ancestor relations are not infer-deleted.

Predecessor-successor linking or Revision, Sequence, and Version number (RSV) linking

During import all the work items are processed for creating predecessor-successor relations to build the successor chain. Following is the RSV linking priority order:

1. For the **StrucBl** class, all objects related to a common master (primary) by the **ItemRev** relation are considered as objects from one series and appropriate **Successr** relation is created between predecessor and successor.

2. Key definition is used whenever it is available to query and link objects in the series.

3. **Sub_label** in GS identity is given the next priority and is used for finding objects in the series when key definition is not available and **sub-label** is populated in the GS identity of the object being imported.

4. A class constant called **EMSAttrListForObjectSeriesC** is defined to hold attribute list and attributes in the list are used to query the objects in the series. This is the last priority. By default, the class constant for the **File** class points to an attribute list that has **WorkingRelativePath** attribute in it.

Propagating lifecycle history information from predecessor to successor

During import of any object, the lifecycle history information from the predecessor is propagated to the object being imported if the predecessor is not frozen. If the predecessor is frozen, the lifecycle history information only from the active lifecycle is propagated. The propagation is dependent on the **Successr** relation created between the predecessor and the object being imported.

The **EMSImpSuccrRelsToPrcPredRelC** class constant is used to control the relation propagation behavior during import. If the class constant is set for any relation class then the relation is considered for propagation. The class constant must point to a value set with **Successr** relations in it. A relation is propagated only if the **Successr** relation created between the predecessor and the object being imported is present in the value set pointed by the class constant.

By default, following are the values of the **EMSImpSuccrRelsToPrcPredRelC** class constant for different relation classes:

```
PrhStr.EMSImpSuccrRelsToPrcPredRelC = "EMSImpSuccrRelsToPrcPrhStr";
where
EMSImpSuccrRelsToPrcPrhStr is defined as
define value set EMSImpSuccrRelsToPrcPrhStr["ChekInR","CkiTeamR"];

PrcHsR.EMSImpSuccrRelsToPrcPredRelC = "EMSImpSuccrRelsToPrcPrcHsR";
where
EMSImpSuccrRelsToPrcPrcHsR is defined as
define value set EMSImpSuccrRelsToPrcPrcHsR["ChekInR","CkiTeamR"];

Sig.EMSImpSuccrRelsToPrcPredRelC = "EMSImpSuccrRelsToPrcSig"; where
```

```
EMSImpSuccrRelsToPrcSig is defined as
define value set EMSImpSuccrRelsToPrcSig["ChekInR","CkiTeamR"];
```

As per these values, the **PrhStr**, **PrcHsR**, and **Sig** classes are considered for propagation when the **Successr** relation between the predecessor and the object being imported is either **ChekInR** or **CkiTeamR**.

For example:

1. Create an assembly, **Asm,A,1** in Teamcenter and export it to Teamcenter Enterprise for reference.
2. In Teamcenter Enterprise, search for **Asm,A,1** and submit it to lifecycle.
3. In Teamcenter, create another assembly, **Asm,A,2** and export it to Teamcenter Enterprise for reference.
4. In Teamcenter Enterprise, search for **Asm,A,2** and verify if **Asm,A,1** is replaced by **Asm,A,2** in lifecycle.
5. Verify **Asm,A,1** is not in lifecycle.
6. Verify that the actors on lifecycle processes are able to work on assignments generated for **Asm,A,2**.

Importing data into Teamcenter Enterprise

Importing data into Teamcenter Enterprise is an indirect action. In other words, there is no user interface to perform the import action. When a user exports data from Teamcenter to Teamcenter Enterprise using the export action, Teamcenter Integration Framework invokes the export API in Teamcenter. On receiving the data from Teamcenter, Teamcenter Integration Framework invokes the data mapper to generate mapped data. Teamcenter Integration Framework then invokes the import API of Teamcenter Enterprise to import data.

Data is present as factors in the import XML file. Each factor contains Teamcenter Enterprise objects.

Each factor in the import XML file is assigned a factor depth by the data mapper. Once invoked, Teamcenter Enterprise importer downloads the XML file using FMS. Before processing the data contained in the XML file, the importer checks for concurrency. If the XML file contains any object that is present in some other XML file being imported, the current XML file is queued and is processed only after the import of the other XML file is complete. GS identity is used to identify the presence of same objects in other XML files.

Teamcenter Enterprise parses the XML file using the **SAX** parser and processes data one factor at a time. The objects in the import XML file are converted to their Teamcenter Enterprise representation and processed starting from factor with lowest depth. For example, factors with depth 0 are processed first, then factors with depth 1 are processed, and so on. Factors with same depth are processed as per their order of appearance in the import XML file.

> Note:
>
> Ownership transfer from Teamcenter to Teamcenter Enterprise is not supported for Teamcenter authored data.

While processing an object for import, it is saved in the database as a new object if it is not available in the database. If it exists in the database, the existing object is updated. The object can also be skipped. Teamcenter Enterprise creates a *transaction log file* that contains a summary table listing all the objects processed as per their element ID in the XML file along with the corresponding Teamcenter Enterprise class name and state (new, update, or skip). A separate table summarizes the reason for skipping objects.

Outdated Teamcenter Enterprise replica objects

Outdated replica icons

Consider that the Teamcenter Enterprise site has a replica object. When a user updates the corresponding primary object in the remote site, the replica at the Teamcenter Enterprise site becomes outdated. An outdated replica object is represented by an icon with two green dots and a red line, as shown in the following example.

Outdated replica object	Icon
Assembly	
Component	
Document	

Check for an outdated replica in the thin client

1. In the **Search Results** pane, choose **More Actions→Check Replica Sync Status**.
2. If the replica object, such as a replica of an assembly, is outdated, it is represented by the outdated assembly () icon.

Check for an outdated replica in the classic client

1. Select the object for which you want to check the replica sync status.
2. Choose **Actions→GMS Actions→Check Replica Sync Status**.

3. If the replica object, such as a replica of an assembly, is outdated, it is represented by the outdated assembly () icon.

Check for an outdated replica from Team Browser

1. Choose **Tools→Search** to search for objects. Select one or multiple replica objects.
2. Choose **Remote Actions→Check Replica Sync Status**.

   All selected replica objects are checked. If a replica object, such as replica of an assembly, is outdated, it is represented by the outdated assembly () icon.

Incomplete Teamcenter Enterprise replica objects

Incomplete replica icons

Consider that an object is exported from Teamcenter to Teamcenter Enterprise for reference with **Transfer Top Assembly** selected as the BOM option in the transfer option set (TOS). In this case, only the parent object in the BOM is sent as a replica whereas the children below the top assembly are sent as stubs.

> Note:
>
> However, if you export the object with **Include Entire BOM** as the BOM option in the TOS, the entire BOM is sent as a replica.

After exporting the object, verify in Teamcenter Enterprise that it is represented by an icon with two gray squares. The two gray squares imply that the object is incomplete, as shown in the following example.

Incomplete object	Icon
Assembly	
Component	
Document	

Check for incomplete objects in Teamcenter Enterprise

1. Create a transfer option set (TOS) named **TopLevelAssm** in Teamcenter with **Transfer Top Assembly** selected as the BOM option.

2. Export an object, such as an assembly, for reference with **TopLevelAssm** as the TOS to an Teamcenter Enterprise site.

   On a successful export, the  icon in Teamcenter Enterprise shows that the assembly is incomplete.

Monitor the export request

1. In the left navigation pane of the thin client, click the **TcGS Monitoring Status** link.

2. Search for the **Activity Status** in Teamcenter Integration Framework. The search result displays the different statuses of an export or import activity. The different statuses are:

   - **Scheduling**
   - **Data Export**
   - **Data Mapping to replica site**
   - **Data Import to replica site**
   - **Confirm Data Export**

A. Standard mapping

Introduction to mapping

When moving data between PDM (product data management) systems, users simply want to see their parts, assemblies, and other associated data as it moves from system to system. PDM systems routinely store data from other applications, however, a universally complete representation for all PDM data in all forms and uses, that satisfies all requirements is not possible.

PDM systems track both data and relationships between different pieces of data. The addition of relationship management complicates the data mapping. Because relationships are important to PDM data and there is no common approach to dealing with relationship management, different systems take different approaches to solving the problem.

The PDM system is often responsible for launching the data into CAD and other authoring applications. Virtually every PDM system has a library of integrations allowing the PDM system to interact with, launch, and effectively store application data for various applications including CAD tools, visualization and simulation packages, word processors, spreadsheets, and presentation packages. A successful translation of PDM data must support all required PDM to application integrations in the receiving system. The need to support these applications adds a layer of complexity to the problem.

There is also a subtle interplay among PDM functionality, the data a business must store, and the underlying business process the PDM system must support. Business processes are designed to exploit the features and strengths of the PDM system. The decision of where to store business data is tightly bound to the functionality the PDM supports. For example, the Teamcenter Enterprise versioning system is so strong that the choices customers make in how to use Teamcenter Enterprise exploits the versioning system. When moving this data into Teamcenter, important decisions must be made in areas where Teamcenter does not have an equivalent concept or does not provide the same kind of functionality. In Teamcenter, one of the few objects that has a guaranteed ID uniqueness is an **Item** object. Many Teamcenter implementation decisions are based on this feature. A successful map of data from one system to another must take this into account.

Teamcenter Enterprise to Teamcenter mapping

Unless otherwise stated, all mappings are invertible. If a Teamcenter **Item** object maps to an Teamcenter Enterprise **ComponentMaster** object, it may be assumed that **ComponentMaster** object maps to **Item** object.

The standard mapping control files map standard data model between Teamcenter Enterprise and Teamcenter. If the data model in your installations is not customized, you can use these control files to map between Teamcenter Enterprise and Teamcenter.

The following table shows selected data model and attributes that are mapped. (If attributes under a class are not shown, they are not mapped.)

Teamcenter Enterprise class	Teamcenter Enterprise attribute	Teamcenter class	Teamcenter attribute
Cmponent		ItemRevision	
	OBID		UID
	OriginalCreationDate		creation_date
	LastUpdate		last_mod_date
	OwnerName		owning_user
	VersionNumber		revision_id
	Revision		revision_id
	InLifeCycleName		Status
CmpsMstr		Item	
	OBID		UID
	LastUpdate		last_mod_date
	OwnerName		owning_user
	PartNumber		item_id
	DefaultUnitOfMeasure		uom_tag
Assembly		ItemRevision	
	OBID		UID
	OriginalCreationDate		creation_date
	LastUpdate		last_mod_date
	OwnerName		owning_user
	VersionNumber		revision_id
	Revision		revision_id
	InLifeCycleName		Status
AssyMstr		Item	
	OBID		UID
	LastUpdate		last_mod_date
	OwnerName		owning_user
	PartNumber		item_id
	DefaultUnitOfMeasure		uom_tag
Document		ItemRevision	
	OBID		UID
	OriginalCreationDate		creation_date
	LastUpdate		last_mod_date
	OwnerName		owning_user

Teamcenter Enterprise class	Teamcenter Enterprise attribute	Teamcenter class	Teamcenter attribute
	VersionNumber		**revision_id**
	Nomenclature		**Name**
	Description		**description**
	Revision		**revision_id**
	InLifeCycleName		**Status**
CADDoc		**Dataset**	
	OBID		**UID**
	OriginalCreationDate		**creation_date**
	LastUpdate		**last_mod_date**
	OwnerName		**owning_user**
	Nomenclature		**Name**
	Description		**description**
	Revision		**rev**
	InLifeCycleName		**Status**
Text data item		**Text** dataset	
	OBID		**UID**
	OriginalCreationDate		**creation_date**
	LastUpdate		**last_mod_date**
	OwnerName		**owning_user**
	Nomenclature		**Name**
	Description		**description**
	Revision		**rev**
	InLifeCycleName		**Status**
JT data item		**DirectModel** dataset	
	OBID		**UID**
	OriginalCreationDate		**creation_date**
	LastUpdate		**last_mod_date**
	OwnerName		**owning_user**
	Nomenclature		**Name**
	Description		**description**
	Revision		**rev**
	InLifeCycleName		**Status**
WordDoc data item		**MSWord** dataset	

Teamcenter Enterprise class	Teamcenter Enterprise attribute	Teamcenter class	Teamcenter attribute
	OBID		UID
	OriginalCreationDate		creation_date
	LastUpdate		last_mod_date
	OwnerName		owning_user
	Nomenclature		Name
	Description		description
	Revision		rev
	InLifeCycleName		Status

Known issues with Teamcenter Enterprise to Teamcenter mapping

Data preservation

The basic intend of mapping data, usually on a field-by-field basis, is data preservation. The simplest mapping is for nonrelationship, nonlist-of-value data. Both Teamcenter Enterprise and Teamcenter have capabilities to extend the data model to allow the storage of arbitrary fields.

The out-of-the-box mapping for the Teamcenter Enterprise **BusinessItem** class maps 20 of 50 available attributes. Only attributes that are important are mapped. Attributes that exist to provide some form of system functionality, and that are never seen or manipulated by the user, are not mapped. Also, if there is a way to achieve the same functionality without mapping the attribute, the attribute is not mapped.

For your site, there may be a unique business process using one of these attributes and the attribute value must be preserved through translation. In these cases, the site administrator must extend the mapping to support the attribute using the Mapping Designer. Any custom attributes have to be supported by extending the mapping, and possibly the receiving system, to support the data. In Teamcenter, this means using the **ExtendedData** or **Form** extension capabilities to add new fields, and in Teamcenter Enterprise, this means adding attributes in the right spot in the class hierarchy. You must then extend the mapping to map the appropriate attribute between the two systems.

In the case where there is a limited set of allowed values (a list of values), the mapping has to be able to provide a table lookup capability to map from one set of allowed values to another.

Preventing data loss

Mapping data between different systems results in the possibility that data can be lost under certain circumstances. An example of this possibility is when the following conditions exist:

- An object is transferred from Teamcenter Enterprise to Teamcenter.
- The object ownership is transferred to Teamcenter.

- The object is subsequently determined to have been transferred in error and the transfer is undone.

This situation could result in the loss of data because when an object is transferred from Teamcenter Enterprise to Teamcenter, some attributes are dropped during the mapping processing. This is due to data model incompatibility and because some Teamcenter Enterprise attributes are not needed in Teamcenter (or vice versa). When ownership is transferred back to Teamcenter Enterprise from Teamcenter, data loss could occur because when an attribute is not sent to Teamcenter, there is no way for it be repopulated in Teamcenter Enterprise on the reverse trip.

Behavior preservation

You must map the data so it behaves correctly when it is sent to downstream applications and upon launch, to the supported tool set. This drives many design decisions. For example, in Teamcenter correctly launching a product structure to a CAD system or VIS application dictates that the CAD files be stored as datasets underneath the item revisions that represent the parts. This requires mapping of Teamcenter Enterprise CAD docs to Teamcenter datasets. Other Teamcenter Enterprise document types are best mapped as document type items.

Many integrations are one-off libraries written to support a single application embedded in a single PDM system. Therefore, critical data in the PDM system used for correctly integrating the application data may be useless in the another PDM system. The mapping to get the correct supporting data in the PDM system may be complex.

Fields may be added to objects in the PDM system to help find the correct data at the correct level of detail. Because different PDM systems have different find functionality with different levels of control and functionality, even a simple field copy must be thought through to make sure the right data is easily visible.

Data in a PDM system is the subject of multiple business processes. Processes are often adjusted to exploit the strengths in the PDM system in use. As data moves into the different PDM system, the mapping must allow the new system's required business process to act on the required data.

Revision/version/sequence

Teamcenter Enterprise **BusinessItem** objects vary by revision, version, and sequence.

Teamcenter **Item** objects have revision and sequence, but do not support versioning. You can reliably identify a Teamcenter part by supplying its **Item** ID and its revision ID. In Teamcenter Enterprise, **PartNumber**, **Revision**, and **VersionNumber** attributes are all needed to resolve to a part that is actually used or released. The loss of version data can be handled in many ways. The version number can be concatenated to the revision ID. The underlying business process can send only the latest version of a revision, or the official version. The actual revision or version may not be used in the Teamcenter Enterprise installation. This needs to be resolved on an installation by installation basis.

Only **Item** objects in Teamcenter have the revision concept (excluding dataset revisions). However, **CADDoc** objects must map to datasets to allow correct data interpretation in VIS and the CAD

integrations. **CADDoc** objects have versions and there is no facility to specify a dataset version independently in the CAD integrations.

The **Official** field is used to indicate which version among the series of versions is the official version. This does not exist in Teamcenter, in part because the one revision is the official one. If this must be stored somewhere, it probably belongs on a primary form. The standard mapping ignores this field. Even if you wanted to preserve this field, you must remember the Structure Manager and the CAD integrations do not discriminate on it.

As a minimum, **PartNumer**, **Revision**, and **Version** data must be maintained. You can always use a naming scheme to preserve this information. One method is to form the **Item** ID using the **PartNumber** attribute concatenated by a period (**.**) with the **Revision** attribute (for example, *partNum.revision*). The Teamcenter revision then becomes the version. Alternatively, the part number can become the **Item** ID and the Teamcenter revision can become the Teamcenter Enterprise revision concatenated with the version number.

Naming uniqueness and conventions

Naming conventions can be used to resolve some issues. However, only a few objects in Teamcenter require unique names. Names of Teamcenter that are transferred to Teamcenter Enterprise may violate the Teamcenter Enterprise name uniqueness rules. You can avoid this by embedding the **Item**/**Revision** IDs into the names, when available. Or you can use a hash method to include the UID into the name. This guarantees uniqueness, but at the cost of readability.

Relationships gap

One of the biggest differences between Teamcenter and Teamcenter Enterprise is how objects relate to each other. In Teamcenter, objects can point to each other or be related through the **ImanRelation** GRM (generic relationship management) object. In Teamcenter Enterprise almost every interobject relationship is represented using the Teamcenter Enterprise relation class. Also, Teamcenter GRMs rarely hold data, whereas in Teamcenter Enterprise, the relation objects have information detailing how the particular object is used in the relationship. In Teamcenter, you typically have another object (for example, a **Workspace** object) that holds the information and then points to the related object.

If an Teamcenter Enterprise relationship has no interesting data on it, for example, the standard **Attach** or the **PartDoc** relation, a GRM is the correct mapping. In other cases, other objects are appropriate, for example, some Teamcenter Enterprise (**CADOccR**, **CfgOccR**, **AssmStct**) objects map to the **PSOccurrence** object in Teamcenter. For other relationships, the data on the relationship must be stored on the primary or the secondary relationship. It depends on the mapping and the assumed cardinality of objects on both sides of a relationship.

Ownership model gap

In Teamcenter, each **pomApplicationObject** object has an owning user and an owning group. In Teamcenter Enterprise, objects may be owned by individuals, teams, vaults or by external systems.

Permission model gap

In Teamcenter Enterprise, rights to data can be assigned on a message-by-message basis. Teamcenter uses a read/write/delete/import/export/project model.

Mapping strategy

Siemens Digital Industries Software recommends that you begin the mapping by assessing the known business processes in the installed system in the field. The processes, or the goals of those processes, must be supported in the new system. Rather than try to map every field and object between the two systems, start by determining how those business processes must be supported in the new system, independent of existing data. If a site relies heavily on the versioning system in Teamcenter Enterprise, determine how these processes will be supported in Teamcenter. This drives the solution for how to bridge the version gap. When going from Teamcenter to Teamcenter Enterprise, determine which messages qualify as a write to effectively resolve the permissions.

Special mapping topics

Teamcenter forms, primary forms, and Teamcenter Enterprise extensions

One major difference between Teamcenter and Teamcenter Enterprise is the way objects are typically extended. In Teamcenter, the easiest way to extend an **Item** or **Revision** object is to create a new subtype of that **Item** and **Revision** object and then define a new primary (master) form for the new type. Depending on the usage, the form attributes can then be made properties of an **Item**/**Revision** object using the Business Modeler IDE compound property feature. You can do this without writing any code.

In Teamcenter Enterprise, attributes are commonly added to objects by inserting objects with the requisite attributes in the class hierarchy and then simply inheriting them. It is also common to define a new **BusinessItem** objects and then relate them to other **BusinessItem** objects through custom relations that store unique business data.

Depending on data usage, the appropriate mapping may be from **MasterForm** object to data fields in the **BusinessItem** class or to some related **BusinessItem** class. And the reverse is true. Do not spend much time on mapping Teamcenter **Form** data into Teamcenter Enterprise. The standard mapping assumes that **MasterForm** data is not important or has a natural mapping into standard Teamcenter Enterprise objects. This is a standard model limitation on both sides. The expectation is that there is a lot of data mapped onto the **ItemRevision** primary forms of various types.

Ownership

In Teamcenter, every **pomApplicationObject** object has an owning user and an owning group. In Teamcenter Enterprise, objects may be owned by individuals, teams, vaults or by external systems.

In use cases where data is being worked on in both Teamcenter Enterprise and Teamcenter, a best practice is to create a user and a group in Teamcenter for each team and vault in Teamcenter Enterprise.

These users and groups should have the same name as the teams and vaults in Teamcenter Enterprise. This allows users that used to looking for data by team to find the objects through an ownership lookup. Also, permissions can be effectively granted and revoked by adding users to groups and removing them from groups. Data owned by a user can be mapped to data owned by the Teamcenter user.

When moving data from Teamcenter to Teamcenter Enterprise, map the owning group to the appropriate team. This makes it easy to find the correct data and allow the correct permissions. When moving Teamcenter data into Teamcenter Enterprise, do not retain individual ownership because this makes it difficult for others on the team to work on it.

This implies that ownership transfer is not totally invertible. If this is not acceptable, you can attempt to develop a mapping to maintain ownership, but it may be very difficult to do this.

Permission models

In Teamcenter Enterprise, permissions are typically granted by team membership. But the Teamcenter Enterprise model allows a fine level of access and permission control by specifying users that can be excluded from an object. Also, users can be granted specific permissions on an object by object and on a method by method basis. In Teamcenter, permissions are granted at the read/write/modify/delete/import/export level of granularity. You cannot grant specific methods to specific users. Therefore, team membership should be used to set up permissions as objects move into Teamcenter from Teamcenter Enterprise. Special access groups can be created, with the appropriate permissions and individuals added to the group as needed.

You should not map the Teamcenter Enterprise **AclUserNames**/**AclUserMsgGrps** objects for performance reasons. Large ACL trees seriously degrade Teamcenter performance. However, if this functionality must be preserved, and if the performance penalty that a complex ACL tree exhibits is acceptable, the Teamcenter Enterprise **AclUserNames** functionality can be mapped. You will have to determine which Teamcenter ACL maps most closely the message in the Teamcenter Enterprise message group. The mapper engine builds up the appropriate ACL and then publishes it to Teamcenter. The standard mapping does not use this technique.

IDS model

In Teamcenter, the IDS model is considered a precise assembly with the potential for absolute occurrence overrides at every assembly level. In Teamcenter Enterprise, a particular instance of a part in the first level assembly is represented by three relationship objects: the **AssmStruc** object that shows quantity and links the parent **Assembly** object to the child's primary component, a **CADOccR** object that identifies the child component, and the **OccCfgR** object that places the location for a given configuration. In Teamcenter, all this functionality is encompassed in the **PSOccurrence** object.

However, in the IDS model, a component is identified in its parent, and in every assembly level up through the assembly hierarchy. These are denoted by **CDAOccR** objects that point to **CADOccR** objects instead of **AssmStruc** objects. Special code in the mapper engine handles this case due to its complexity. The standard mapping maps these correctly, allowing them to be repositioned at locations other than in their parents' location in the overall assembly.

Any customizations on the **BusinessItem** class are not impacted by the assembly information and are mapped normally. Customizations on the **CADOccR**, **OccCfgR**, or other IDS relation objects can be stored in occurrence notes if they are not references to other objects. If they are references to other objects, they can be stored as attachments in the attachment windows.

Datasets, documents, and data items

To enable CAD tools to access parts correctly, the Teamcenter Enterprise **CADoc** object has to map to the Teamcenter **Dataset** object. To determine the correct dataset type, you use the **DataItem** tool, making this a many to many mapping.

This means that there is the potential for loss of information because the **CADDoc** and the **DataItem** objects both have information and the mapping forces a less rich set of data fields. If needed, the **Dataset** object can have an associated **Form** object to store extra data.

When moving data from Teamcenter to Teamcenter Enterprise, some **Dataset** data is duplicated in the **CADoc** and the **DataItem** objects.

There may also be an identity issue when mapping the **Dataset** object to the **CADoc** object. Teamcenter datasets have no guarantee of name uniqueness. Often Teamcenter Enterprise **Doc** and **CADoc** objects have a defined key. In these cases, the **Dataset** object, **ItemRevision** object, and item IDs can included in the **Dataset** name to guarantee uniqueness. Alternatively, the **Dataset** UID can be concatenated to the name. However, this creates a unique but not a very readable solution.

When dealing with a non-CAD Teamcenter Enterprise **Document** object, it is better to map it to the Teamcenter **ItemRevision** object of type **Doc**. However, an **ItemRevision** object must have a unique ID and an Teamcenter Enterprise **Document** does not require uniqueness. However, there usually is a uniqueness criterion on a **Document** object and that can be used for the item ID.

Standard mapping rationale

Mapping Teamcenter Enterprise business items to Teamcenter items

Virtually every Teamcenter Enterprise **BusinessItem** object maps to a Teamcenter **ItemRevision** object. All Teamcenter Enterprise **BusinessItem** objects have a status and can be visible to the user and targets of a workflow. In Teamcenter, this makes them **Workspace** objects. All Teamcenter Enterprise **BusinessItems** objects are versionable and the best practice for versionable objects is to make them some type of an **ItemRevision** object.

Some fields do not appear on the mapped list nor on the ignored list. This is because some fields are supported in the mapping but not as a direct field. For example, the **ItemRevision** type attribute is not mapped directly, but the type is supported because it is used to determine which **BusinessItem** object to map to.

Teamcenter Enterprise relationships and Teamcenter

In Teamcenter Enterprise, relationships are used to relate **BusinessItem** objects to each other and to **DataItem** objects. These relationships are usually the only way objects relate to each other. In Teamcenter, objects are related through a GRM on occasion, but often objects either point to each other directly or contain a list of objects that may themselves point to other objects. For example, Teamcenter **ItemRevision** objects point to their parent **Item** objects directly through the **items_tag** attribute. In Teamcenter Enterprise, **Component** objects reference their primary (master) components through the **TcEnt::ItemRev** relationship.

Relationships are often used to drive applications and often exist solely to support a specific application or business process. Consequently, the most important objective in mapping relationships is to ensure the correct application logic is applied on the translation on the other side.

Unlike the **BusinessItem** object mappings, there is no generalization that can be made about the Teamcenter Enterprise relationships, other than, if you are not sure what to do, you can always create a **TC_reference** relationship or other general relationship.

Teamcenter Enterprise data items and Teamcenter

In Teamcenter Enterprise, **DataItem** objects represent data that is stored in bulk outside the relational database. This includes CAD files and other bulk data. The Teamcenter equivalent is the **ImanFile** object. However, because **DataItem** objects may be released, they must map to at least a Teamcenter **Workspace** object. Therefore, the **DataItem** object maps to a **Dataset** object.

There are some subtle differences in the two systems. For example, some CAD **DataItem** objects hold the tool used for the data, but in many others the tool is assumed. Also, the tool is associated to the **DataItem** object. In almost all cases, when you encounter an Teamcenter Enterprise **DataItem** object you create a **Dataset** object and possibly an **ImanFile** object. When moving data from Teamcenter into Teamcenter Enterprise, at least some **Dataset** data and the **ImanFile** information is mapped to the **DataItem** object.

Standard Foundation mapping

The following are a sampling of factors in the Foundation mapping from Teamcenter Enterprise to Teamcenter. The factor names indicate the mapping.

Factor	Description
OwnedItm_to_WorkspaceObject	Maps **OwnedItm** to a **WorkspaceObject**.
BusItem_to_ItemRevision	Inherits from **OwnedItm_to_WorkspaceObject** factor and maps a **BusItem** to **ItemRevision**.
StrucBI_to_ItemRevision	Extends **BusItem_to_ItemRevision** and maps **StrucBI** to **ItemRevision**.
DataItem_to_Dataset	Extends **OwnedItm_to_WorkspaceObject** and maps a **dataitem** to a **dataset**.

Factor	Description
File_to_MISC	Extends **DataItem_to_Dataset** and maps a **File** to **MISC**.
ItemMstr_to_Item	Maps **ItemMstr** to **Item**.
AssmStrc_to_PSOccur_PSAlter	Maps **AssmStrc** to either **PSOccurrence** and/or **PSAlternateList**.
AssmEff_to_Effectivity	Maps **AssmEff** to **Effectivity**. Handles both unit and date effectivities.
RevEff_to_Effectivity	Maps **RevEff** to **Effectivity** object. Handles both unit and date effectivities.
Attach_to_TC_Attaches	Maps **Attach** relation to **TC_Attach** relation.
BitToVis_to_IMAN_Rendering	Maps **BitToVis** relation to **IMAN_Rendering** relation.
BusItem_add_ReleaseStatus	Maps **BusItem LifeCycleState** to appropriate **ReleaseStatus**.
OwnershipMap	Maps **ownername** and **EMSSiteId** to **User** class.

Standard CATIA mapping

There are two projects for CATIA for each transfer direction, one for BOM and one non-BOM transfers. A complete CATIA solution is based on Foundation plus, CATIA BOM, and CATIA non-BOM mappings. The following table lists sample factors for the transfers from Teamcenter Enterprise to Teamcenter.

BOM factors

Factor	Description
x0CatPrt_to_CATPart_CATIA	Inherits from Foundation's **File_to_MISC** factor and changes the inherited primary target of **x0CatPrt** to **CATPart** Type.
x0CatPrd_to_CATProduct_CATIA	Inherits from Foundation's **File_to_MISC** factor and changes the inherited primary target of **x0CatPrd** to **CATProduct** Type.
x0CTMod_to_catia_CATIA	Inherits from Foundation's **File_to_MISC** factor and changes the inherited primary target of **x0CTMod** to **CATIA** Type.
g2AsmPos_to_PSOccurrence_-imprecise_CATIA	Inherits from Foundation's **AssmStrc_to_PSOccur_PSAlter** factor and populates the **PSOccurrence** properties from **g2AsmPos**, including transformation matrix.
x2AsmPoQ_to_PSOccurrence_-imprecise_CATIA	Inherits from **g2AsmPos_to_PSOccurrence_imprecise_CATIA** factor and for each **x0TrafoListRow in x2AsmPoQ.x0TrafoList**, creates a secondary target object as **PSOccurrence**. Populates properties including transformation matrix.
Attach_GenDoc_g0GenBin_to_IMAN_-specification_CATIA	Inherits from Foundation's **Attach_to_TC_Attaches** and changes the inherited primary target of **Attach** to **IMAN_specification** and overrides the **primary_object** to be the target of the **Part**.

Factor	Description
x0CatDrw_add_ItemRevision_SA_CATIA	Creates secondary target objects of **x0CatDrw** as **ItemRevision** with accompanying item, primary (master) forms, and **IMAN_master_form_relations** whenever the root of the **x0CatDrw WorkingRelativePath** (file name) does not match the **PartNumber** of the **Part** with which it is associated.
Attach_GenDoc_x0CatDrw_to_IMAN_-specification_CATIA	Inherits from **Attach_GenDoc_g0GenBin_to_IMAN_specification_CATIA**. For **x0CatDrw** considered as secondary, that is, when the file name root of the **x0CatDrw.WorkingRelativePath** does not match the **PartNumber** of the **Part** with which it is associated. Under these circumstances a new **ItemRevision** is created corresponding to the **x0CatDrw**, this factor sets the **primary_object** of the inherited primary target of **Attach** to be that new **ItemRevision**.

Non-BOM factors

Factor	Description
x0CatPrt_add_ItemRevision_NB_CATIA	For **x0CatPrt** considered as standalone, that is, when exists a **x0CatPrd** associated with the same **Part** as **x0CatPrt**, creates secondary targets of **x0CatPrt** as **ItemRevision**, **ItemRevision-Master**, **IMAN_master_form** (for IR/IRM), **Item**, **Item--Master**, and**IMAN_master_form** (for I/IM).
x0CatPrd_to_CATProduct_CATIA	Inherits from **Attach_GenDoc_x0CatPrt_add_-IMAN_specification_CATIA**. For **x0CatPrt** considered as standalone, that is, when a **x0CatPrd** is associated with the same **Part** as **x0CatPrt**, under which circumstances a new **ItemRevision** is created based on the **x0CatPrt**. This factor sets the **primary_object** of the inherited primary and secondary targets of **Attach** to the new **ItemRevision**.
x0CatPrt_add_PSOccurrence_NB_CATIA	For **x0CatPrt** considered as non-BOM, that is, when the **x0CatPrt.GSIdentity.label** (generally corresponds to OBID) is listed as a value in **x0NewModelObids** of the **x0CatPrd** associated with the same **Part** as **x0CatPrt**, creates secondary targets of **x0CatPrt** as **PSOccurrence** and **PSOccurrenceNotes**.
GenDoc_add_Folder_Non-BOM_Parts_NB_CATIA	Creates a Non-BOM parts folder and corresponding relations and references when **x0CatCgr** is present, and both **x0CatPrt** and **x0CatPrd** exist on the same **GenDoc** or multiple **x0CatPrt** objects exist on the same **GenDoc**.
GenDoc_add_Folder_ShapeReps_-NB_CATIA	Creates a **ShapeReps** folder and corresponding relations and references, and remove existing relations based on **Attach** (between target of **Part** and target of **x0CatCgr**) when **x0CatCgr** is present, and both **x0CatPrt** and

Factor	Description
	x0CatPrd exist on the same **GenDoc** or multiple **x0CatPrt** objects exist on the same **GenDoc**,

Standard Pro/ENGINEER mapping

A complete Pro/ENGINEER solution must be based on Foundation plus Pro/ENGINEER mappings. To distinguish Pro/ENGINEER factors from Foundation factors (because both are present in the mapper project Factor directory), all Pro/ENGINEER factors end in **_PROE**. The following table provides a sample list of factors for transfers from Teamcenter Enterprise to Teamcenter.

Factor	Description
GenAssm_add_is_precise_PROE	Sets the **is_precise** attribute to **"1"**, on all **BOMView—Revision** objects mapped from Teamcenter Enterprise **Assembly** objects. (The Foundation factor that creates the full **BOMView—Revision** object always sets **is_precise="0"**.)
ProAMemR_ProAsm_to_PSOcc_-parent_PROE	Creates a **PSOccurrence** with **parent_bvr** pointing to the BVR created from the **Assembly**, without a **child_item attr**. Use the **ProAMemR_ProObj_to_PSOcc_child_PROE** factor, to set the **child_item** because it starts from the **ProAMemR.Right** and traverses to the **Part** object.
ProAsm_ProSRep_to_ipem_SR_-Names_PROE	Set the **ipem_SR_Names attr** on the **ProAsm** object (**ProSrepR.Right**) by appending the **ProSrep**(s) (**ProSrepR.Left**) **BrowserWindowDesc** attribute value(s) to it.
ProAsm_to_ipem_MetaDataForm_-PROE	Sets **ipem_SR_Names=DEFAULT REP** for **ProAsm** only.
ProAsm_to_ProAsm_PROE	This child factor of the **File_to_MISC** Foundation factor sets the **ref_names** attribute to **"AsmFile, MetaData"**. Similar factors set **ref_names** to appropriate values for **ProDrw**, **ProFrm**, **ProPrt**, and so forth.
ProAsmI_to_ProAsm_PROE	For **ProAsmI**, sets **ref_names=MetaData** and **ref_types=2** only.
ProDepnd_ProObj_to_IPEM_-dependency_PROE	Creates **IPEM_dependency** and **IMAN_specification** objects. The **IPEM_dependency** is created as if the **WorkingRelativePaths** of the two **ProObj-s** are different (the actual comparison is done in the child factors due to a Mapforce limitation). The **IMAN_specification** is created as if they are the same, but is turned off with **GMSRemoveFromProcessing**.
ProDepnd_ProPrt_ProDgm_to_-IPEM_dependency_PROE	This child factor of **ProDepnd_ProObj_to_IPEM_dependency_PROE** that compares **WorkingRelativePath** attributes of a related **ProPrt** and **ProDgm** pair of objects, and makes the appropriate adjustments, if necessary, to the **IPEM_dependency** and **IMAN_specification** objects created in the parent factor. Similar factors exist for

Factor	Description
	ProPrt and **ProFrm**, **ProPrt** and **ProRep**, and so forth. and also for **ProAsm** and **ProDgm**, **ProAsm ProFrm**, and so forth.
ProDMdlR_ProPrtI_to_new_-Items_PROE	Creates the **Item**, **ItemRevision** objects and so forth for the **ProDrw** and creates an **IMAN_spec** connecting the new **ItemRevision** to the **ProDrw** when all of the following are true: • The **WorkRelativePath** attributes of the **ProPrtI** and **ProDrw** objects do not match and **ProDrw** is not a Stub. • There is not an **Attach** relation connected to the **ProDrw** (the **ProDrw** may have its own **Part** attached to it, and we do not want to handle such a case in this factor). A similar factor, **ProDMdlR_to_new_Items_PROE**, exists for a **ProObj** (**ProPrt** or **ProAsm**) on the **ProDMdlr.Right**.

B. Teamcenter Data Exchange concepts and components

Exporter and importer

The exporter and importer components provide serialization and deserialization of transferred data. In the serialization process, the exporter follows TC XML schema to generate XML fragments and creates export records for objects exported to a target site.

Supported objects

The following objects are supported by the exporter and importer:

Teamcenter base objects:

- **Alias**
- **Dataset**
- **Form**
- **IdContext**
- **Identifier**
- **ImanFile**
- **ImanRelation**
- **Item**
- **ItemRevision**

Teamcenter product structure objects:

- **AbsOccData**
- **AbsOccDataQualifier**
- **AbsoluteOccurrence**
- **APN**
- **AssemblyArrangement**
- **AssemblyArrangementAnchor**
- **PSBOMView**
- **PSBOMViewRevision**
- **PSOccurrence**
- **PSOccurrenceNotes**
- **VariantCondition**
- **VariantExpression**
- **VariantOption**
- **VariantRule**

Teamcenter site objects:

- **ADA objects**
- **ImanExportRecord**
- **ImanReservation**
- **POM_stub**
- **PublicationAuditRecord**

The following objects are not supported, but the references to these objects from supported Teamcenter objects are exported through properties. These objects must be defined at the importing site using the Business Modeler IDE.

Teamcenter base objects:

- **Project**
- **ReleaseStatus**
- **Tool**
- **UnitOfMeasure**

Teamcenter organization objects:

- **Group**
- **GroupMember**
- **Person**
- **POM_imc**
- **Role**
- **User**

Teamcenter product structure objects:

- **Effectivity**
- **PSNoteType**

The importer supports transfer of the following CATIA objects from Teamcenter Enterprise:

- Basic document-centric classes
- V5 CGR files
- V4 and V5 drawing files
- Hybrid models, coexistence V4 and V5 models
- Design tables
- Imprecise assembly structures
- Alternate shape representations

The importer supports transfer of the following Pro/ENGINEER objects from Teamcenter Enterprise:

The exporter supports transfer of CAD neutral files and metadata to Teamcenter Enterprise.

Parts (assemblies/components) that have documents without any attachments are imported into Teamcenter as empty datasets without any named references.

The following CAD data is not supported for transfer by the importer or exporter:

- Absolute occurrences
- Options and variants
- Effectivities
- Precise CATIA assembly structure
- Imprecise Pro/ENGINEER assembly structure

Supported transfer options

Option group	Supported option	Symbol name	Value
Item Options	**Include All Revisions**	**opt_all_revs**	true | false
	Latest Revision Only	**opt_revs**	Latest
	Latest Working Revision Only	**opt_revs**	Latest Working
	Latest Working/Any Release Status	**opt_revs**	Latest Working and Released
	Latest Any Release Status	**opt_revs**	Latest Released
	Selected Revision(s) Only	**opt_revs**	Selected
	Specific Release Status Only	**opt_revs**	Released status
Dataset Options	**Include All Versions**	**opt_all_ds_vers**	true | false
	Include All Files	**opt_all_ds_files**	true | false
Product Structure Options	**Include Entire BOM**	**opt_entire_bo**m	true | false

Option group	Supported option	Symbol name	Value
	Transfer Top-Level Item Only	**opt_do_struct**	true | false
	Exclude Transfer-Protected Components	**opt_trans_prot_comp**	true | false
	Exclude Export-Protected Components	**opt_exp_prot_comp**	true | false
General Options	**Exclude Export-Protected Objects**	**opt_exp_prot_obj**	true | false
	Exclude Folder Contents	**opt_folder_contents**	true | false
Relations	**Relations to be included**	**opt_grm**	Supported through closure rule
	Include JT datasets	**opt_jt_ds**	true | false
	Include NX datasets	**opt_nx_ds**	true | false

Scoper

When the exporter or importer receives a request to transfer data, it determines the transfer mode from the option set object. It then creates the traversal object based on the transfer mode scope (import, export) and the schema format (PLM XML or TC XML). The basic behavior of a traversal object is to provide two methods, **process** and **traverse**. The **process** method processes the present object in hand and calls the **traverse** method. The **traverse** method navigates to the next object and then calls the **process** method. This operation iterates until no more objects are available to process.

The scoper evaluates the transfer mode closure rules and the options provided by the exporter and returns only the objects that are identified for **process** in a closure rule clause. This list is sent to the exporter or importer for serialization or deserializaton.

Data mapper

Product life cycle management systems track product object data and relationships. To move data from one system to another, an object's attributes and its relationship to other objects must be converted or mapped to the representation in the other system. For moving data from one Teamcenter system to another, you only need to map customizations as the standard Teamcenter object attribute and relationships are the same in both systems.

Teamcenter provides a standard mapping process for moving data between Teamcenter Enterprise and Teamcenter systems. This standard mapping does not map all object attributes. Attributes that are not used in the destination system are not mapped. Attributes that are used only to provide system

functionality that can be achieved through another method and are not displayed to the user, are also not mapped. Due to customizations to support a unique business process, you may need to map some of these standard attributes.

Reference elements can be mapped to multiple objects and will accept any number of mappings. A reference element must map to a string that matches the string of characters prepended by **plm:** for the object that it is mapped to.

For a **GSIdentity** object, the class attribute is information only and not the class name in the receiving system.

Data synchronizer

The data synchronizer uses *recipes* defined by initial replication to determine if replicated data is up-to-date or needs to be updated. You can use the synchronization functionality to see if data that is replicated on your site is outdated and update it as desired. You can also push data that has changed to sites where replicas exist. The synchronization function cannot be used to update the following:

- Meta model data such as types
- Administrative data such as users, groups, roles, and business rules
- Security information such as Access Manager (AM) rules

Candidates for synchronization are determined by comparing the primary object's last modification date to the replica's last export date. Through a customization, you can also identify candidates automatically providing an event trigger that calls an API that flags replicated objects as out-of-date.

A *recipe* is a combination of import export options that is encoded as transfer formula strings. This transfer formula is stored as part of the **ImanExportRecord** object as a string identifying the name of the transfer formula and an integer field for the associated transfer formula variables presented in the form of a map. The map contains a key string and a value string. This allows generalized behavior of these options. The following options that can be used to form the transfer formula:

- **All Item Revisions**
- **Revision Rule**
- **Release Status**
- **Export File**
- **Selected Revision Only**
- **All Datasets Version**

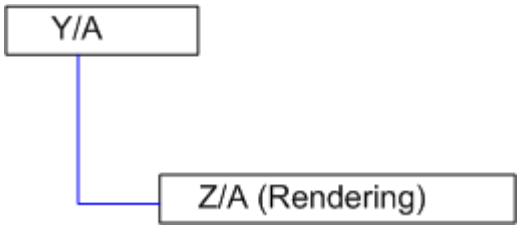
- **Included Generic Relationship Manager (GRM)**
- **Excluded GRM**

Recipes for individual components are stored, but when components are imported in multiple assemblies, only the last recipe is stored, as shown in the following use cases:

Component in two assemblies with differing relationships

1. User A, at site 2, replicates assembly **X** (which includes component **Z**) from site 1 with the **include_relation** value set to **Manifestation**.

2. User B, at site 2, imports assembly **Y** (which also includes component **Z**) from site 1 with the **include_relation** value set to **Rendering**.



   The recipe is overwritten and item revision **Z/A** has the **Rendering** relationship.

3. User A deletes replica assembly **Y** or **Z**.

   Item revision **Z/A** has the **Rendering** relationship.

Synchronization with a released and a working item revision

1. User A, at site 2, imports item **X** from site A with revision selector set to **Latest Released**.
2. User B, at site 2, imports the same item **X** with revision selector set to **Latest Working**.
3. Item revision **X/B** is released and then revised to **X/C**.
4. Item revision **X/C** is released and then revised to **X/D**.

5. User B, at site 2, synchronizes item **X** with site 1.

   The **Latest Working** revision (**X/D**) is replicated to site 2, because the last exported recipe is **Latest Working**.

Synchronization with a released and a deleted working item revision

1. User A, at site 2, imports item **X** from site A with revision selector set to **Latest Released**.
2. User B, at site 2, imports the same item **X** with revision selector set to **Latest Working**.
3. Item revision **X/B** is released and then revised to **X/C**.
4. Item revision **X/C** is released and then revised to **X/D**.
5. User A deletes replica revision **X/B** at site 2.
6. User B, at site 2, synchronizes item **X** with site 1.

   The **Latest Working** revision (**X/D**) is replicated to site 2.

Identity manager

Data Exchange uses the **GSIdentity** object to provide a unique identity for items that are moved between systems that have their own identity mechanism. Due to differences in data models and the intended item's use, the **GSIdentity** object may be unique from the system ID for the item, such an OBID or UID. Some objects may have several equivalent objects in the destination systems. Therefore, the **GSIdentity** object for the exported object may equate to multiple objects in the destination system that must reflect the original **GSIdentity** attributes.

The **GSIdentity** class is a subclass of the **POM** class and contains the following attributes:

- **system**

  Provides a token indicating the data origination, for example, Teamcenter Enterprise or Teamcenter.

- **label**

  Contains a textual representation for the data's UID or OBID in its native system.

- **sub-label**

  Contains the UID or OBID of helper objects in many-to-one mappings.

- **class**

Contains the calling class name.

- **split_token**

  Contains information the translator uses to identify the individual components for one-to-many mappings.

- **context**

  Tracks the attribute name when an attribute in one system becomes the full object in another system. It may also identify (usually as an index) a runtime object that becomes a persistent object in the receiving system. This occurs when the runtime object has uniqueness only in the context of a persistent object. In many instances, neither of these two cases apply and this attribute is empty.

- **factor_id**

  Contains a unique string that identifies a group of Teamcenter Enterprise objects that belong together.

- **atomic**

  Provides a token that indicates whether the label is simply the UID or OBID of an object or something more complex.

The **GSIdentity** object is persisted and associated with the transferred object created in the new system. Objects have only one identity. The composition of **GSIdentity** data provides a key that remains consistent and unique for the data's life or may be changed systematically if the mapping algorithm is altered. The **collect_garbage** utility can be used to remove **GSIdentity** objects from the database for objects that have been deleted.

Ownership manager

The ownership manager controls ownership of objects that are transferred among systems and participates in import and export operations. When a part is processed for transfer of ownership, ownership of all part revisions is transferred to the receiving site. A transfer of ownership can be for the top-level parts only or for an assembly and all its child parts. If ownership of a replicated object is transferred to a new site, the export records for the object are transferred to the receiving site so that the owning site is always aware of existing replicas and all replicas are updated with the new site ownership data.

When ownership of an object that is published to an ODS is transferred to another site, the ODS publication record is updated with the new owning site.

During the ownership transfer process, objects at the exporting site are locked to prevent modifications and are flagged as in the **Ownership Transfer Pending** state. This flag is cleared after the exporting site receives a message from the importer that the transfer is complete and the exporting site marks the objects it has as replicas. If the ownership transfer process terminates unexpectedly, the ownership may

be in an inconsistent state. The **ensure_site_consistency** utility allows you to report objects that are in the **Ownership Transfer Pending** state and to correct inconsistent ownership.

The following is an example process for exporting an object with transfer of ownership:

1. The user selects objects to be exported from site 1 to site 2.
2. The user selects the option to transfer ownership.
3. The exporter processes the objects to be exported, for example, remove objects with no export privilege, add helper objects, process closure rules, and so forth.
4. The ownership manager processes this list and removes objects in workflow, removes shared objects, adds additional objects to maintain ownership consistency, for example, all item revisions for an item.
5. The exporter adds export records for the objects to the list of objects.
6. The ownership manager locks the objects from further modification.
7. The exporter processes the object list and generates a TC XML file.
8. The importer at site 2 imports the objects from the TC XML file including export records.
9. The ownership manager sets the site ownership for imported objects as site 2. Objects imported as replicas are left out.
10. The importer keeps track of objects for which transfer of ownership is successful.
11. Site 2 generates a TC XML file with success/failure state for ownership transfer for each object and sends it back to site 1.
12. Site 1 reads the objects in the TC XML file and sends a request to the ownership manager to unlock them.
13. The ownership manager unlocks all objects from this list and also changes the **owning_site** attribute for objects whose ownership was successfully transferred.
14. The ownership manager sends a message to Object Directory Service (ODS) regarding ownership changes.
15. Teamcenter Integration Framework sends messages to sites where replicas and stubs are located regarding ownership changes.
16. The ownership manager deletes the export records for these objects.

Siemens Digital Industries Software briefcase

Briefcase functionality primarily provides a method of exchanging information with suppliers or other sites that do not share a live connection with your enterprise. It allows data to be replicated and also provides the capability to check out data to a site for modification (the owning site remains the original site, but the remote site can change objects, for example, add components to the structure, change geometry, and so forth). briefcase data is packaged in an archive file. The TC XML file generated by the exporter and all the data files referenced in the TC XML file are packaged into a briefcase file. This file can then be imported at the receiving site to create the replica objects and files.

A site can be defined as a hub (when configuring sites). A hub site can replicate replica objects to other sites, typically when delegating work to sub-contractors. If an item is checked out to a hub site, the hub site can also check out that item to another remote site for modification.

Objects checked out to a site are subject to the following modification privileges:

- All users at the site can change the object subject to Access Manager (AM) rules, provided no user at the site has performed a local object checkout.
- If a user performs a local object checkout, only that user can change the object.
- If an object is checked out to remote site, users at the local site cannot change the object.

The following is the process Teamcenter uses for packaging a briefcase file:

1. Export the data files referenced in the TC XML files to the transient volume.
2. Update the TC XML files to contain the relative path to the data files instead of FMS file tickets.
3. Create the briefcase file using the TC XML data and the data files, and include the following information in the briefcase file as attributes:
    - Exporting site
    - Target site
    - List of TC XML files
    - Teamcenter version
4. Create a briefcase dataset object with the briefcase file and log file attached to it as named references, and attach the briefcase dataset object to a Teamcenter mail that is sent to the exporting user.

Briefcase also can be used to schedule an export through Teamcenter Integration Framework, allowing the export to proceed in the background so that you can immediately gain rich client session control for other purposes. The following is the process for a scheduled export:

1. Teamcenter calls an SOA service to initiate the offline export.

2. Teamcenter Integration Framework schedules the offline export.

3. Teamcenter Integration Framework locates a server at the exporting site and calls and SOA service to export the objects. The TC XML files and log files are generated after the export.

4. If mapping is required, Teamcenter Integration Framework calls the mapper to translate the TC XML files.

5. Teamcenter Integration Framework locates a server at the exporting site and calls the package briefcase contents SOA service to create the briefcase file.

6. If the exporting user chooses to receive e-mail notification after export, Teamcenter Integration Framework sends an e-mail notification to the user. This e-mail contains information about the briefcase dataset object created and its URL after the export.

7. Teamcenter calls the export API to export the objects. The TC XML files and log files are generated after the export.

8. Teamcenter calls the package briefcase API to create the briefcase file.

The briefcase feature does not support transfer of ownership.

File Management System file transfer

Data Exchange uses File Management System (FMS) to transfer data files between Teamcenter PLM systems. FMS transfers files on demand and uses the FMS server cache (FSC) to improve file transfer performance. FMS keys are stored in the PLM system databases for use by the server processes to generate tickets for a specific system. FSC servers use keys read from key files referenced in their configuration files.

For Teamcenter, default keys are set in the Teamcenter database and can be modified using the **install_encryptionkeys** utility. FMS in Teamcenter uses these default keys unless you configure it to use key files referenced in the **FMSmaster.xml** configuration file.

FMS in Teamcenter Enterprise does not provide default keys; they are explicitly set during the install process. You must also configure the FMS key files.

For information about using FMS with Teamcenter Enterprise, see the Teamcenter Enterprise *Network and Database Configuration Guide*.

Security Services

Security Services allows a user to sign on one time for access to multiple Teamcenter products. Teamcenter Data Exchange support this functionality when its components are properly configured to use single sign-on (SSO).

C. Standard conditions, message groups, participants, and subscription

Standard Data Exchange conditions

The following is the standard condition provided with Data Exchange:

ExportProtectedC

Determines if an object can be exported to a remote site. If the condition is set to **true** for an object; the object cannot be exported to a remote site.

AllowReqstObjCreC

If the condition is set to **true**, a request object is persisted in the database. The object can then be submitted to a life cycle and at the last step of the lifecycle, it will be submitted to Teamcenter Integration Framework.

Standard Data Exchange user groups

Group	Description
GMS_OT_Export_UsrGrp	Allows users to give the ownership of an object to a remote site.
GMS_GSToTCE_UsrGrp	Allows users to perform Teamcenter Enterprise actions from Teamcenter Integration Framework.
GMS_Export_UsrGrp	Allows users to perform a remote export transaction.

Standard Data Exchange message access rules

For more information about rules, see the *Teamcenter Enterprise Administrator's Manual*.

> Note:
>
> All standard Data Exchange rules have a positive child flag; that is, all the subclasses of the specified class inherit the rule.

Participant	Message/ Message Group	Class	Condition	Description
GMS_Export_UsrGrp	**GMS_Export_MsgGrp**	**PdmRoot**	**TRUE**	Allows the users of the **GMS_Export_UsrGrp** user

Participant	Message/ Message Group	Class	Condition	Description
				group to export objects for reference.
GMS_OT_Export_-UsrGrp	**GMS_OT_Export_MsgGrp**	**PdmRoot**	**TRUE**	Allows the users of the **GMS_OT_Export_UsrGrp** user group to export objects for ownership transfer.
GMS_OT_Export_-UsrGrp	**Query_grp**	**Switch**	**TRUE**	Allows the users of the **GMS_OT_Export_UsrGrp** user group query for a switch.
GMS_OT_Export_-UsrGrp	**Owner_Access_grp**	**Switch**	**TRUE**	Allows the users of the **GMS_OT_Export_UsrGrp** user group to attach objects to or detach objects from a switch.
GMS_OT_Export_-UsrGrp	**GetObjectsForSwitchD**	**Switch**	**TRUE**	Allows the users of the **GMS_OT_Export_UsrGrp** user group to get objects attached to a switch.
GMS_GSToTCE_UsrGrp	**GMS_GSToTCE_MsgGrp**	**PdmRoot**	**TRUE**	Allows the users of the **GMS_GSToTCE_UsrGrp** user group to invoke messages available in the **GMS_GS_ToTCE_MsgGrp** message group.
admin grp	**GenerateXMLSchema**	**PdmRoot**	**TRUE**	Allows the users of the **admin grp** user group to generate the XML schema.
admin grp	**PopulateEMSContextOBID**	**PdmRoot**	**TRUE**	Allows the users of the **admin grp** user group to populate the **EMSContextOBID** attribute for objects that are imported using the **PopulateEMSContextOBID** command line message.
admin grp	*****	**Switch**	**TRUE**	Allows the users of the **admin grp** user group to perform any action on a switch.
admin grp	**Create_rel_grp**	**TmToCRC**	**TRUE**	Allows the users of the **admin grp** user group to create a relation between a transfer mode and closure rule clause.
admin grp	**Modify_rel_grp**	**TmToCRC**	**TRUE**	Allows the users of the **admin grp** user group to modify the relation between

Participant	Message/ Message Group	Class	Condition	Description
				a transfer mode and closure rule clause.
admin grp	Delete_rel_grp	TmToCRC	TRUE	Allows the users of the **admin grp** user group to delete the relation between a transfer mode and closure rule clause.

Standard Data Exchange message groups

Message group name	Message name
GMS_Export_MsgGrp	UnimplementedMsg
GMS_Export_MsgGrp	ExportItem
GMS_Export_MsgGrp	ExportItemDP
GMS_Export_MsgGrp	UpdateReplicaStatus
GMS_Export_MsgGrp	ValidateForExport
GMS_Export_MsgGrp	ValidateForReplicas
GMS_Export_MsgGrp	DoExport
GMS_Export_MsgGrp	ExportAsReplica
GMS_GSToTCE_MsgGrp	UnimplementedMsg
GMS_GSToTCE_MsgGrp	CheckReplicaSyncState
GMS_GSToTCE_MsgGrp	ConfirmExport
GMS_GSToTCE_MsgGrp	ExportObjects
GMS_GSToTCE_MsgGrp	GetSynchronizeList
GMS_GSToTCE_MsgGrp	ImportObject
GMS_GSToTCE_MsgGrp	SynchronizeToSite
GMS_GSToTCE_MsgGrp	UpdateObjectsOnOwnershipChange
GMS_GSToTCE_MsgGrp	updateObjectsOnOwnershipChange
GMS_GSToTCE_MsgGrp	UpdMstrObjsOnReplDel
GMS_GSToTCE_MsgGrp	UpdMstrObjsOnReplToStb
GMS_GSToTCE_MsgGrp	UpdMstrObjsOnStubRepl

Message group name	Message name
GMS_OT_Export_MsgGrp	UnimplementedMsg
GMS_OT_Export_MsgGrp	GMS_Export_MsgGrp
GMS_OT_Export_MsgGrp	ExportForOwnership

D. Standard Data Exchange item classes

These are the item classes provided with Data Exchange when it is initially installed. Your site may have created additional classes.

Item class	Displayed as	Description
ClosRule	**Closure Rule Clause**	Defines the rules for gathering objects during the export action.
EMSevtQ	**Enterprise Multi-Site Event Queue**	Specifies a queue of events that are processed for synchronize action.
ExpRec	**Export Record**	Records details of export of a business item or a data item to a remote site. The exporter creates only one export record per object per site. An export record is created only for objects and not for relationships.
ExprtReq	**Export Request Object**	Captures the user request for data export.
ExRecRel	**Export Record Relation**	Specifies a relation between a business item or a data item and its export record.
ExpSpill	**Exporter Spill**	Specifies the temporary store class for spilled objects during export.
GMSCfg	**GMS Configuration**	Records the configuration variables required by Teamcenter Enterprise Data Exchange.
IdentLst	**Identity List**	Specifies the identity list for objects that are imported.
ImpRec	**Import Record**	Records details of a business Item or a data item imported from a remote site.
RelImpRc	**Import Record Relation**	Specifies a relation between a replica item and its import record.
Site	**Site**	Represents a data sharing end point in the Global Multi-Site network.
Stub	**Remote Object**	Specifies a placeholder representation of a real object.
StubRel	**Remote Relation**	Specifies relations involving stubs.
Switch	**Switch**	Groups objects to be flipped for ownership transfer.
SyncObj	**Sync Object**	Maintains details of a business item or a data item to be synchronized.

Item class	Displayed as	Description
TranMode	**Transfer Mode**	Specifies a logical grouping of closure rule clauses.
TransOpt	**Transfer Option Set**	Specifies a set of transfer options that are used for import or export operation.
TransTrk	**Export Transaction Tracking Object**	Tracks the export transactions.
TmToCRC	**Transfer Mode To Closure Rule Clause Relation**	Specifies a relation that associates a closure rule clause with a transfer mode.

E. CATIA Teamcenter Enterprise Integration classes

Standard CMI item classes

Class	Description
x0Assm(assembly)	Specifies the CATIA assembly.
x0AsmMstr(AssmMstr)	Specifies the assembly base (master).
x0CTMod	Specifies the CATIA model.
x0CatPrt	Specifies the CATIA part file.
x0CatDrw	Specifies the CATIA drawing file.
x0CatCgr	Specifies the CATIA CRG shape representation.
x0CatRep	Specifies the persistent class that stores CAT V5 representations.
x0CatPrd	Specifies the CATIA product file (V5).
x0CatExc	Specifies the design tables that are stored in the **.xls** file.
x0CatTxt	Specifies the design tables that are stored in the **.txt** file.
x0CatPrc	Specifies the CATIA process file.
x0PrdDoc(g0PrdDoc)	Specifies the persistent product document class representing CATIA product document.

Standard CMI relation classes

Class	Description
g2AsmPos	Specifies the assembly structure with position.
x2AsmPoQ	Specifies the assembly structure with position and multiple quantity.
g2PrdDoc	Specifies the **PartDoc** subclass that relates a part and describing document.
x2DepDes	Specifies the CATIA data file and design table dependency relation.

F. Classic Client Pro/ENGINEER integration classes

Standard Classic Client Pro/ENGINEER item classes

Class	Description
ProAsm	Models a Pro/ENGINEER assembly or interchange group.
ProDgm	Specifies a Pro/ENGINEER diagram.
ProDrw	Specifies a Pro/ENGINEER drawing.
ProFrm	Specifies a Pro/ENGINEER format.
ProLay	Specifies a Pro/ENGINEER layout.
ProMfg	Specifies the Pro/ENGINEER manufacturing file that references a part or an assembly model and a model of a tooling fixture used for manufacturing.
ProMrk	Specifies the markups that are created for parts (includes objects with **.prt** extension), assemblies (includes objects with **.asm** extension), drawings, layouts, reports, and diagrams in Pro/ENGINEER.
ProPrt	Models a piece part, composite, or sheet metal part.
ProRep	Specifies the report file.
ProSketch	Specifies the Pro/ENGINEER sketch.
ProSRep	Specifies the simplified representation (collection).
ProPrtI	Specifies a named row in the Pro/ENGINEER family table inside a generic Pro/ENGINEER part.
ProAsmI	Specifies a named row in the Pro/ENGINEER family table inside a generic Pro/ENGINEER assembly.

Standard Classic Client Pro/ENGINEER relation classes

Class	Description
ProDepnd	Relates a Pro/ENGINEER data to another Pro/ENGINEER data that is dependent on it.
ProAMemR	Relates an assembly or assembly instance to its components.

Class	Description
ProDMdlR	Relates a drawing with the models that it references.
ProSRepR	Specifies the association between an assembly and a simplified representation.

G. Data Exchange with Teamcenter Enterprise Change Management

Installing Data Exchange in an existing Teamcenter Enterprise Change Management environment

When you install Data Exchange in an Teamcenter Enterprise installation environment where Teamcenter Enterprise Change Management is already installed, the installation fails while running the **MODeL** command. As a workaround, you can update the **ecmems.prd** and **rqmems.prd** files to regenerate the data model.

> Note:
>
> Skip this topic if you do not use the Teamcenter Enterprise Change Management solution.

Update the production files to generate the data model definition

1. Open the **ecmems.prd** file from the *MTI_ROOT***meta\ems** directory and replace all instances of **Reason** with **EMSReason**.

2. Open the **rqmems.prd** file from the *MTI_ROOT***meta\ems** directory and replace all instances of **reason** with **rqstreason**.

   > Note:
   >
   > Replace the backslash with a slash if you use a Linux system.

3. Run the Configuration Editor. It displays a message prompting you whether to continue with the previous action, **Add solution – EMS**. Click **Yes**.

4. The Configuration Editor prompts you with the **Start from scratch?** message. Click **No**.

   > Warning:
   >
   > If you click **Yes**, the installation fails as the changes you have made to the **.prd** files are overwritten.

H. Verifying and troubleshooting the transfer configuration

Verify the FMS configuration

1. Use the **fscadmin** utility to check the FMS access of your sites. For example:

```
fscadmin -s http://TcEntHost:4444 -k FMS_ent.key ./status
FSC id: FSC_TcEntHost, site:TcEntHost
 running for 3 days, 17 hours, 3 min, 46 sec
Current number of file connections: 0
Current number of admin connections: 1
fscadmin -s http://cvg101:4001 -k FMS.key ./status
FSC id: FSC_TcHost1, site:CVG101
 running for 0 days, 7 hours, 5 min, 16 sec
Current number of file connections: 0
Current number of admin connections: 1
fscadmin -s http://cvg102:4001 -k FMS.key ./status
FSC id: FSC_TcHost2, site:CVG102
 running for 0 days, 5 hours, 7 min, 51 sec
Current number of file connections: 0
Current number of admin connections: 1
```

2. If an FSC for a site is not running, check the FMS configuration using the **fscadmin** utility against the configuration requirements. For example:

```
fscadmin -s http://cvg101:4001 -k FMS.key ./config
```

Verify the transfer from Teamcenter

1. Log on to My Teamcenter as a user with Data Exchange export privileges.

2. Search for or create an item to transfer.

3. Select the item check box and choose **Tools→Export →Remote using Global Multi-Site**.

4. In the **Export To Remote Site using Global Multi-Site** dialog box, if the default site is not the desired destination, click the select target site button  . In the **Select Target Site** dialog box, select the site listed in the **Selected Sites** list and click move left ◀, select the destination site from the **Available Sites** list, click move right ▶, and click **OK**.

5. In the **Export To Remote Site using Global Multi-Site** dialog box, select the **Immediate** check box and click **OK**.

Verify the transfer from Teamcenter Enterprise

To verify the configuration of an Teamcenter Enterprise Data Exchange site, use the thin client to transfer a replica to a Teamcenter Data Exchange site.

1. Log on to your Teamcenter Enterprise thin client as a user with Data Exchange export privileges.

2. Search for or create an item to transfer.

3. Select the item and choose **More Actions→Remote Export**.

4. In the **Remote Export Options** pane, select the check box for each item that you want to transfer and click **OK**.

5. Select the destination site from the **Site** list and the desired transfer option set from the **Transfer Option Set** list.

6. Select the **Immediate** and **Continue On Error** check boxes and click **Finish**.

   The thin client displays a message indicating the export was successful.

Troubleshooting transfer process errors

Transfer process diagnostics

1. A user initiates a transfer from the remote import function of the rich client by:

   a. Selecting the item to export.

   b. Setting the transfer options.

   c. Selecting the destination site.

   d. Clicking **OK**.

   No Teamcenter Integration Framework activity is displayed at this point.

2. Teamcenter sends a request for objects.

   a. Teamcenter sends a SOAP message to Teamcenter Integration Framework requesting the objects from Teamcenter Enterprise.

      No Teamcenter Integration Framework activity displayed at this point.
      Diagnostic action

- Check the Teamcenter server **syslog** file to see if the SOAP message was created.

  Location: By default, **C:\temp**.

  Name: **tcserver**<*nnnnnnnn*>**.syslog**

  Information: Search for a **sig 11** or **sig 22** error which indicates the SOAP message was not created.

- Check the Teamcenter Integration Framework activity status to confirm the SOAP message was received.

- Check the Teamcenter Integration Framework log for errors.

- Check the rich client for error messages.

  Location: By default, **%Temp%**

  Name: <*user*>**_session**<*nnnnnnnn*>**.log**

b. The Teamcenter Integration Framework process creates a request object.

Teamcenter Integration Framework activity: **Scheduling**.

The **Status** column indicates the activity is **In Progress** if it is scheduled for off hours; otherwise the status is **Complete**.

Diagnostic action

- Log on to Teamcenter Integration Framework and search for activity status using the **Message ID** or other known search criteria.

  To display more information about the activity from the Teamcenter Integration Framework log, or if you suspect a Teamcenter Integration Framework failure, you can check the Teamcenter Integration Framework application log file. To access this log file:

  A. Log on to Teamcenter Integration Framework as an administrator (**IFAdmin** by default)

  B. Click **View** →**Activity Status**.

c. Teamcenter Integration Framework passes the request to Teamcenter Enterprise through a SOAP message.

Teamcenter Integration Framework activity: **Data Export**.

The **Status** column indicates **In Progress**

Diagnostic action

- Check the Teamcenter Integration Framework activity status to ensure the message was sent.
- Ensure the SOAP message contains the transfer formula and transfer option set.
- Check the Teamcenter Enterprise server log to ensure the request message was received and the export started.
- Check the Teamcenter Integration Framework log for errors.
- Check the Teamcenter Enterprise server console and verify the MUX connection exists.

3. The Teamcenter Enterprise scoper generates the TCE XML file with metadata.

Teamcenter Integration Framework activity:	**Data Export**.
	The **Status** column indicates the activity is **In Progress**.

Diagnostic action

- Check the Teamcenter Enterprise server log for errors.

4. The data exporter outputs the file to the Teamcenter Enterprise FMS transfer location.

Teamcenter Integration Framework activity:	**Data Export**.
	The **Status** column indicates the activity is **In Progress**.

Diagnostic action

- Check for the TCE XML file in the FMS transfer location.
- Check the Teamcenter Enterprise server log for errors.
- Check the FMS log for errors.

5. FMS 2 generates an FMS read ticket and Teamcenter Enterprise passes it to Teamcenter Integration Framework with notification of the completed export.

Teamcenter Integration Framework activity:	**Data Export**.
	The **Status** indicates the activity is **Complete**.

Diagnostic action

- Check the Teamcenter Integration Framework activity status to ensure the ticket arrived.
- Check the Teamcenter Integration Framework log for errors.

6. The XML data file is transferred to Teamcenter Integration Framework for access by the data mapper.

Teamcenter Integration Framework activity:	**Data Mapping**. The **Status** indicates the activity is **In Progress**.

Diagnostic action

- Check for a temporary file in the Teamcenter Integration Framework transfer directory.

7. Data mapper maps data using the map control file (MCF) to Tc XML format.

Teamcenter Integration Framework activity:	**Data Mapping**. The **Status** column indicates the activity is **In Progress**.

Diagnostic action

- Check data mapper log for errors.
- Check the Teamcenter Integration Framework log for errors.

8. Teamcenter Integration Framework transfers the TC XML data to the FMS transient volume.

Teamcenter Integration Framework activity:	**Data Mapping**. The **Status** column indicates the activity is **Complete**.

Diagnostic action

- Check for FMS transfer.

9. Teamcenter Integration Framework notifies Teamcenter that mapping is complete, and FMS 1 provides a ticket to transfer the data.

Teamcenter Integration Framework activity:	**Data Import**. The **Status** column indicates the activity is **In Progress**.

Diagnostic action

- Check the Teamcenter Integration Framework log for errors.

10. FMS 1 transfers the TC XML file to Teamcenter

Teamcenter Integration Framework activity:	**Data Import**. The **Status** column indicates the activity is **In Progress**.

11. The Teamcenter data importer parses the TC XML file and creates the metadata in the database.

Teamcenter Integration Framework activity:	**Data Import**.
	The **Status** column indicates the activity is **In Progress**.

Diagnostic action

- Check the Teamcenter **syslog** file for errors.

12. FMS 2 (Teamcenter Enterprise) transfers the bulk data files to the Teamcenter volume.

Teamcenter Integration Framework activity:	**Data Import**.
	The **Status** column indicates the activity is **Complete**.

Diagnostic action

- Check the FMS log for errors.

13. Data is imported (not shown in the).

Teamcenter Integration Framework activity:	**Confirm Export**.
	The **Status** column indicates the activity is **Complete**.

Diagnostic action

- Check Teamcenter and verify the parts are imported successfully.
- Check Teamcenter Enterprise and verify export records are created for the objects.

Determine transfer activity status

You can use the Teamcenter Integration Framework log files to get information about a transfer, and you can use the Teamcenter Integration Framework activity status to help you locate information about a transfer. You use the Teamcenter Integration Framework **Site Configuration and Monitoring** configuration interface to help you track the transfer process.

This procedure is based on using an activity status business object definition.

1. Log on to your Teamcenter Integration Framework application as the Teamcenter Integration Framework administrator (**IFAdmin** by default) and choose **Configure→Sites and Monitoring**.

2. Choose **View.→Activity Status**.

3. If the activity of interest is not in the **Search Results** pane, click the expand button  at the left side of the pane to display the **Search Filters** pane and click **Search**. Special wildcard characters are supported. Click **Special Characters supported** at the bottom of the pane to display them.

4. Type filter values into the available boxes to narrow the search to the desired activity and click **Search**.

   Teamcenter Integration Framework displays a table containing a row for each transfer activity.

5. Select the failed activity and record the last six digits if the **Message ID** (includes a dash between the digits). For example, in the following result ID, the last six digits are **229–884**.

   ```
   11877229677229–884
   ```

   These digits provide enough information to uniquely identify the transfer activity.

6. Select and copy the content in the **Message Status** column for the record.

   This information indicates the action and options for this activity, which helps you understand the process.

7. Determine the step that has failed and record its start time. This time can be used to help locate information about the failure in the Teamcenter Integration Framework log files.

Download the Teamcenter Integration Framework log file

You can use the Teamcenter Integration Framework user interface to download a log file copy that you can view and search with any text editor. By default, rolling appenders are used to name these log files. These log files contain only the Teamcenter Integration Framework specific messages and are compressed into a single ZIP file that you can download.

1. In the Teamcenter Integration Framework interface, choose **Developer Info→View Default Log File**.

2. In the **Opening LogSnapShot-** *file-id***.zip** dialog box, select **Save File** and click **OK**

You can use any ZIP file tool to extract the log files and view them with a text editor.

I. Troubleshooting Data Exchange

Error logs

You may encounter different errors while transferring data between Teamcenter Enterprise and remote sites. There are two files, also called error logs, that log the errors encountered while working with the Data Exchange solution. The two files are the transaction log and the dispatcher output file. The transaction log contains the summary of the current transaction and is sent through the Teamcenter integration. The dispatcher output file resides at the server side and the level of information contained in this file depends on the value of the debug level of the GMS configuration object.

Every error encountered has an error number associated with it. The **mfail** error code in the error logs records this error number.

Transaction log

The transaction log contains a summary of every transaction, such as exporting data to or importing data from a site. The transaction log for export contains information about the:

- Local and remote sites.
- Transfer formula used for the transaction.
- Root objects used for object gathering.
- Table summarizing the number of objects of different classes along with the intent of export.

When the debug level is set to **Low** or any value higher than **Low**, the factors, closure rules in the factors, and objects gathered using the closure rules are logged in the transaction log.

Every object that is imported is logged in the transaction log in a summary table along with the status mentioning whether is it an import of a new object, update of an existing object, or skip. In case the object is skipped, another table is created in the transaction log mentioning the reasons for the skip.

In case of an error, the transaction log also notes the error code. The transaction log is sent to Teamcenter Integration Framework. When a transaction fails, the information stored in the transaction log helps the user to identify the exact cause of the failure. Detailed information to debug the error is written in the dispatcher output file.

The transaction log is written into the vault location of the site where the data is exported to or imported from. Every transaction log has a number appended to it. This number is the result ID of the transaction and is available on the activity status pane of the Teamcenter Integration Framework configuration interface.

Example:

You exported an object from Teamcenter Enterprise to Teamcenter. After performing the export action, you find that the transaction status in the activity status in Teamcenter Integration Framework is **Halted**. To debug the error that halted the export transaction, you must locate the transaction log and check the error code mentioned in it.

Locate the transaction log path

1. In the thin client Administration Editor, search for the site.
2. On the properties page of the site object, note the vault location name mentioned in the **Vault Location Name** box.
3. Type the vault location name on the search page and click **OK**.
4. The properties page of the vault location gives the path of the transaction log in the **Full Path** box.

Dispatcher output file

The dispatcher output file contains a detailed report of an error that is encountered while performing a transaction, such as exporting data to or importing data from a site. The administrator can refer to this file to debug the error. This file resides on the server. The level of information contained in this file depends on the value of the debug level on the GMS configuration object.

For example, if the value of debug level is **Low**, a minimal set of information is logged in the dispatcher output file. When the debug level is **Medium**, in addition to the information logged for **Low** debug level, the names of the methods used from EMS module during any action are logged. If the value debug level is **High**, along with the information logged for the medium debug level, method arguments and their values, sets of objects and strings, SQL statements, and so on are logged.

Run a script to extract transaction details

To troubleshoot an error, you can extract transaction details from the dispatcher output file by running the **extract_gms_trans.bat** (Windows) or **extract_gms_trans** (Linux) script as follows:

```
extract_gms_trans transaction_id [dispatcher_output_file]
```

Debug levels

To get more information about the errors in the dispatcher output file, you can set debug levels. These debug levels are cumulative. The higher the debug level, more information is written in the error.

The following table lists the different debug levels.

Value	Description
None	No information is logged in the dispatcher output file if the transaction is successful.
Low	Minimal information is logged. For an export action, the factors, closure rules in the factors, and the objects gathered using the closure rules are logged in the transaction log.
Medium	Along with the logging information for the **Low** debug level, method enter and exist messages are logged in the dispatcher output file.
High	Along with the logging information for the **Medium** debug level, method arguments and their values, sets of objects and strings, SQL statements, and so on are logged in the dispatcher output file.

Setting debug levels

If the GMS configuration object is created in the setup, debug loggings are controlled by the **Debug Levels** and **Switch Debug Level to High On Error** attributes of the GMS configuration object. Otherwise, you set the debug levels using the **EMSDEBUG** configuration variable. You must configure this variable explicitly. If you do not set a value for this configuration variable, the value is set to **NONE** by default.

When the GMS configuration object is not present, another configuration variable, named **SWITCH_EMSDEBUG_TO_HIGH_ON_ERR** decides how much debugging information must be written in the dispatcher output file when an error occurs. By default, the value of this configuration variable is **FALSE**. If the value is **FALSE** and an error occurs, the debugging information is not written in the dispatcher output file. If you set the value of this configuration variable to **TRUE**, on error, high-level debugging information is written in the dispatcher output file and the error code in written in the transaction log.

Sample trace

The following is a sample trace for the **GetObjects** message with method tracing (**-tm**) switched on in the **config.cfg** file and the debug level set to **High** in the GMS configuration object. When the debug level is set to **High**, the value of all the arguments are printed during the entry to and exit from any message implemented in the EMS module.

For more information about EMS module messages, see the *API Reference* in the Teamcenter Enterprise help.

Note:

The arguments are not printed by **GetRelatedObjectsWSql**, **QueryDbObjsRelWithExp**, **GetRelClassesToQrySet**, and **GetRelClassesToQry** messages because these messages do not have implementation in EMS module.

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 11  GetObjects @ Scoper
(ScoperGetObjects)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}    Message Enter | <Scoper:GetObjects>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Message | <GetObjects>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[1] |
<input:MTIstring> <className = Scoper>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[2] | <input:
ObjectPtr> <thisObj displayed
below>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ActualEffDate : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ActualRelDate : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ApplicationOwner : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     AssemblyType : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     BaseRevision : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CategoryName : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CheckOutOwner : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CheckedOut : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class : <Assembly>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CreationDate :
<2011/09/09-09:28:32:803>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Creator : <super user>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CurDbName : <sum70b>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DbFetchDate :
<2011/09/09-10:01:40:659>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DisplayedName :
<dam_sep_9_asm_01,A,1,1>
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     dsgLockedForSign : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     g5ThumbImageURL : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0Addressee : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0Author : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0CCAddressee : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DateReceived :
<2011/09/09>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DateTimeReceived : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DocCreateDate :
<2011/09/09>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[3] | <input:
ObjectPtr> <closureRuleObj
displayed below>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class : <ClosRule>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class1ForeignKey : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class2ForeignKey : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ClosureRuleName :
<ItemRev2>
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     SkipSecObjects : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     TransferOption : <>
```

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     UpToDateStatus : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ZBlob : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[4] | <output:
SetOfObjects*>
<factorLinkSet = NULL>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[5] | <output:
integer*> <mfail = 0>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 12  GetRelatedObjectsWSql @
ItemRev
(RelationGetRelatedObjectsWSql)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 13  QryDbObjsRelsWithExp @
ItemRev
(RelationQryDbObjsRelsWithExp)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 14  GetRelClassesToQrySet @
ItemRev
(RelationGetRelClassesToQrySet)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 15  GetRelClassesToQry @
Assembly
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 14  mfail = 0
(RelationRelationGetItemClass)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 13  mfail = 0
(RelationQryDbRelsObjsWithExp)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 12  mfail = 0
(RelationGetRelatedObjectsWSql)
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Message | <GetObjects>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[1] | <input:
MTIstring> <className = Scoper>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[2] | <input:
ObjectPtr> <thisObj
displayed below>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ActualEffDate : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ActualRelDate : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ApplicationOwner : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     AssemblyType : <>
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     dsgLockedForSign : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     g5ThumbImageURL : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0Addressee : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0Author : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0CCAddressee : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DateReceived :
<2011/09/09>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DateTimeReceived : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     tx0DocCreateDate :
<2011/09/09>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[3] | <input:
ObjectPtr>
<closureRuleObj displayed below>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class : <CloseRule>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class1ForeignKey : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class2ForeignKey : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ClosureRuleName :
<ItemRev2>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CreationDate :
<2010/11/02-13:16:42:739>
```

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CurDbName : <admm70b>
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     RelationType : <EMSNonSpan>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     SkipSecObjects : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     TransferOption : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     UpToDateStatus : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ZBlob : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[4] | <output:
SetOfObjects*>
<factorLinkSet size = 2>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     index 0 | ObjectPtr
displayed below
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class : <AssmMstr>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CostMasterOBID : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CreationDate :
<2011/09/09-09:28:32:934>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Creator : <super user>
.
.
.
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Recycled : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     UpToDateStatus : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ZBlob : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     index 1 | ObjectPtr
displayed below
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class : <ItemRev>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class1 : <AssmMstr>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class2 : <Assembly>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Class3 : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CreationDate :
<2011/09/09-09:28:32:963>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     CurDbName : <sum70b>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DbFetchDate :
<2011/09/09-10:01:40:680>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DbName1 : <sum70b>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DbName2 : <sum70b>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     DbName3 : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     EMSProcessInstruction : <4>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     EMSSiteID : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     GlobalDomain : <c6dcda60-
e67c-11df-b1b7-b9f9e68c91cc>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     InterdomainFlag : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     IsPrimaryObj : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     IsRootObj : <+>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     LastUpdate :
<2011/09/09-09:28:32:963>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Left : <vjjjmzcarh---
sum70b--aVQ>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     LifeCycleState :
<LcsWorking>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Nid : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     OBID : <vjjjmzgarh---
sum70b--aVQ>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Recycled : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     RefCopied : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Revision : <A>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Right : <vjjjmyearh---
sum70b--aVQ>
```

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Third : <>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     ZBlob : <->
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}     Argument[5] | <output:
integer*> <mfail = 0>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}    Message Exit | <Scoper:GetObjects>
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140} 11   mfail = 0
(ScoperGetObjects)
```

In the sample trace, a line similar to:

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}   Message Enter |
<Scoper:GetObjects>
```

specifies entry to the message. After that all the arguments are printed.

Argument[*arg_num*] specifies the start of the argument, where *arg_num* is the argument number starting from 1.

When the argument is of type **SetOfStrings** or **SetOfObjects**, each member is printed beginning on a separate line containing **index** *index* provided the number of member in a set does not exceed 1000. *Index* specifies the index of the member in the set starting from 0.

During exit, all the arguments in the message are again printed followed by the message exit line:

```
2011/09/09-15:31:40 pni6w1063 omfsvr <8444> {9-9-2011_153140}   Message Exit |
<Scoper:GetObjects>
```

In the sample trace:

- The first argument is of type **input** with data type as **MTIstring**. The name of the argument is **className** and the value is **Scoper**.
- The second argument is the assembly object for which the closure rule expansion is being performed.
- The third argument is the closure rule object with **ClosureRuleName** as **ItemRev2**.
- The fourth argument, **factorLinkSet**, is the output to hold the gathered objects. The value of this argument is **NULL** at the message entry. At the message exit, this argument contains **AssmMstr** and **ItemRev** objects.
- The fifth argument is for holding the value of **mfail**.

Troubleshoot error: Ownerdb not found

If the data transfer from Teamcenter to Teamcenter Enterprise fails due to a missing owner object, check whether:

- There is a corresponding owner object (vault or team) with the same name as that of the exporting user at the Teamcenter site.
- There is a team or vault location for the objects.
- The importer user has an author role.

For more information about creating a vault, team, and vault location, see the Teamcenter Enterprise *Administrator's Manual*.

Troubleshoot error: Missing required attributes

When transferring data from Teamcenter to Teamcenter Enterprise, if some of the required attributes of the importer are missing, check the mapping between Teamcenter and Teamcenter Enterprise. There is a mapping component in Teamcenter Integration Framework, where the actual mapping is done between the two systems.

Troubleshoot error: Keytbl conflict found

Data transfer from Teamcenter to Teamcenter Enterprise fails when a local object with the same key name but different global stable identity GSID already exists. To resolve this conflict, change the key attributes either on the local object or on the imported object.

Troubleshoot error: OBJSERV timed out

Following is a sample entry for the **omfsrv** service in the **config.cfg** file:

```
"omfsvr" "1 1 5 10 30 60 $(@FILEPATH:q $(PDM_BIN:q) objserv) -C 250"
```

To avoid the **OBJSERV** timed out errors, ensure that you set the **Max** and **Var Max** service settings to the same value. Also, ensure that the value is set high enough to support your workload. In the following example, these values are set to 50.

```
"omfsvr" "1 1 5 50 50 60 $(@FILEPATH:q $(PDM_BIN:q) objserv) -C 250 "
```

Troubleshoot error: OS_SERV timed out

Following is a sample entry for the **OS_SERV** service in the **config.cfg** file:

```
"OS_SERV" "0 0 1 9 3 1 $(OLAUNCHEXE:q) -f $(@FILEPATH:q $(PDM_BIN:q)
osserv)"
```

To avoid the **OS_SERV** timed out error, ensure that you set the **Max** and **Va Max** service settings to the same value. Also, ensure that these values are double the values of the **Max** and **Var Max** service settings of the **omfsrv** service. In the following example, these values are set to 100.

```
OS_SERV" "0 0 1 100 100 1 $(OLAUNCHEXE:q) -f $(@FILEPATH:q $(PDM_BIN:q)
osserv)"
```

Team Browser remote icons unavailable

If the Team Browser remote icons are disabled or unavailable, sync the **.jar** files related to Team Browser at the client end with the files at the server end. The files at the server end are located at *MTI_ROOT***evista\java\classes**. Also, check whether the **ENABLE_EMS_OPERATIONS** and **ENABLE_REPLICA_BEHAVIOR** configuration variables are set to **1**.

Reference out action in Team Browser fails

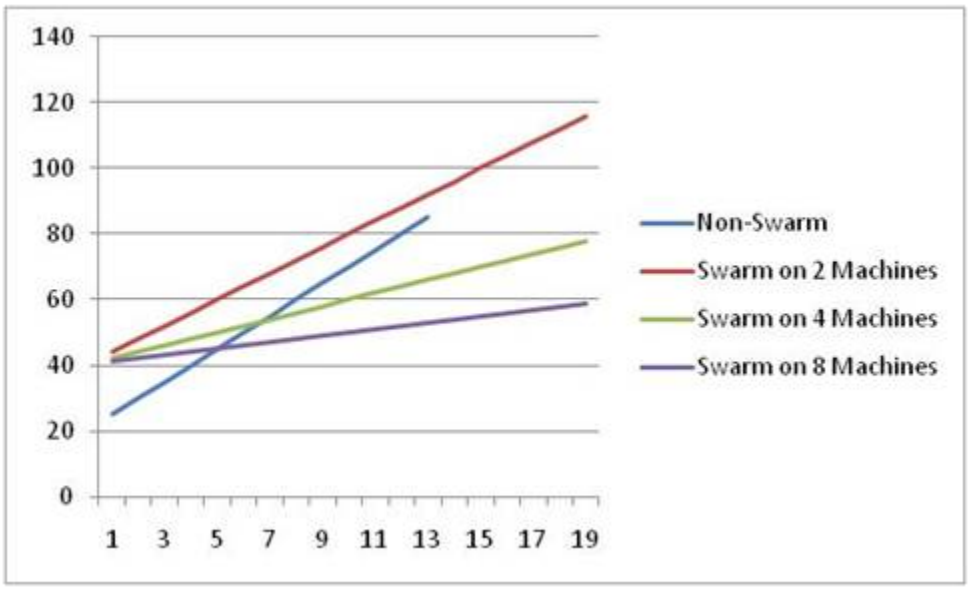
If the reference out action in Team Browser fails for parts created in NX and imported to Teamcenter Enterprise using Global Multi-Site (GMS):

- Check whether the JT file is attached to the NX parts.
- Check whether the **IDSUseCadNeutral** configuration variable is set to **1** in Teamcenter Enterprise.
- Ensure that the transferred data is not corrupt.

J. Data Exchange performance tuning

Swarming

Swarming in Data Exchange is the process of breaking up large data transfer operations into many smaller data transfers that can be processed simultaneously. The benefit of swarming is that it allows for additional hardware to be used to speed up throughput through parallel processing. Swarming changes the shape of the processing time curve for transfers. The following graph provides an example of the potential performance improvement.[1]



Activation of swarming is managed in the exporting system, which determines the upper limit of the size of each transfer composing the swarm. This *upper limit* is usually referred to as the chunk size.

Assume that this graph shows the performance curves for swarming in a particular Data Exchange environment. The X-axis shows the size of data to be transferred, and the Y-axis show the time it takes. The graph shows us:

- When only two machines are available to handle a transfer, swarming improves performance only if the chunk size is 13 or larger. Because increased fixed costs make swarming slower over this range, swarming should be used only when machine limits (out of memory and so forth) are a factor.

- When four machines are available[2] for transfer, a swarming chunk size of seven or larger can improve performance. This allows smaller transactions to benefit from lower fixed costs, while allowing swarming to support faster speeds at higher transfer sizes.

1 The graph shows a concept and does not reflect specific swarming metrics. Therefore, units of measure have been omitted.
2 What available means is important to understand. It means that the resource is not dedicated to a different task at the time. Machine availability is basically a ratio of machines to current tasks.

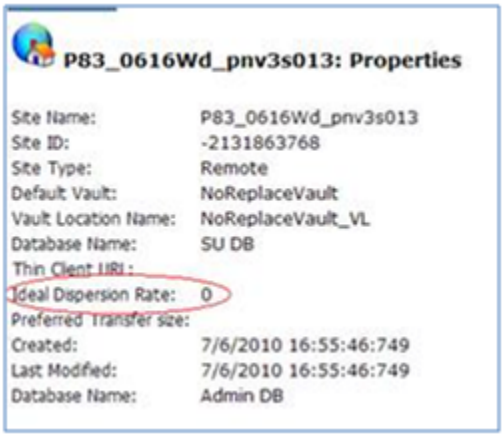
- When 8 machines are available for large transfers, you can guarantee anything within a desired size range (1-20 in the diagram) can be transferred within a time of 60. The chunk size should be set to 5 or larger to maximize performance.

Swarming can be leveraged only for transfers using the continue on error (CoE)[3] option. This limitation exists because all-or-nothing transfers are required to operate within a single session.

Ideal dispersion rates and coalescence

For swarming transfers, breaking up a large transfer into smaller chunks can result in large numbers of *remnant transfers* which are very small transfers required to complete a swarming transaction. Dispersion rate and ad hoc coalescence provide a mechanism to prevent degrading swarming transfer performance due to remnant transfers.

To combat flooding the system with large numbers of small transactions, Data Exchange provides an ideal dispersion rate (IDR) configuration in the exporting system. This configuration limits the number of child transfer requests generated by a parent request.



Teamcenter Enterprise Admin web dispersion rate setting

The value of IDR has the following meaning;

If (IDR == 0), swarming is disabled.
If (IDR > 0), swarming is enabled.

When IDR = N where N > 0; IDR indicates the maximum number of concurrent export requests that can be initiated within a single export transaction. Setting a value for this attribute for a local site has no effect.

3 The basic concept for CoE is that the unit of atomic transfer is an *island of data,* which is a set of objects that together form a logical concept. In CoE Mode, individual islands of data can fail to transfer, but will not affect referential integrity because stub objects are used as place holders for the missing data. If referential integrity for a swarm member cannot be maintained via stubs, then the whole member transfer will fail, which is manageable since references in other swarm members to the failed data will be represented as stubs.

When running Data Exchange on limited hardware, setting IDR = 1 usually gives the best results. When running Data Exchange on horizontally scaled hardware, setting IDR = 2 provides more transactions that Data Exchange has to handle. A higher value of IDR means more parallelism for the Teamcenter Enterprise exporter, which is desirable. Setting IDR above 3 may provide a small additional improvement in performance and can result in flooding Data Exchange with a lot of small transactions.

> Note:
>
> After attempting to get more parallelism in Data Exchange by increasing IDR to 2 or 3, decrease the chunk size. This results in more parallel transactions and keeps the tree balanced.

Ad hoc coalescence means taking a large number of small transfers and combining them together. Unlike ideal dispersion rates, ad hoc coalescence can be used with unrelated initial transfers.

As an example, assume that a site chooses to initiate Data Exchange transfers for each item that is checked into a vault in Teamcenter Enterprise and the transfer formula used for the transfer results in a small number of objects being included. For Data Exchange transfers, the resource consumption is:

$$Cost_{transfer} = Cost_{fixed} + Cost_{size}$$

In this example, the fixed costs can dominate and lead to sub-optimal system performance. Because size related costs are basically linear, combining multiple small[4] requests provides better overall throughput.

> Note:
>
> Data Exchange does not have an implementation of ad hoc coalescence in this release.

Improving performance of large data transfers in Teamcenter Enterprise

Based on the data involved in Data Exchange transfers, many relations can participate in creation of EMS module-specific relations. For example, **ExRecRel**, **RelImpRc**, and **StubRel** relations become extended relationship objects (EROs) in Teamcenter Enterprise. To provide a significant performance increase in the Data Exchange actions, you can add the **NID** attribute as an index on Teamcenter Enterprise relations along with these EMS module-specific relations. Review the data exported from and imported to **Teamcenter Enterprise** and index the **NID** attribute in the applicable database tables.

Teamcenter Enterprise logging configuration

The Teamcenter **EMSDEBUG** variable, located in the **config.cfg** file, is used only for problem debugging. It can have a significant impact on performance and scalability. If left at verbose settings, the resulting

4 To be combined, requests must be sufficiently similar. The similarities have to include, but might not be limited to; source system, destination system, transfer formula, and importing/exporting user (which might or might not be the requesting user, depending on the Data Exchange configuration.)

log files can consume disks. A setting of **PROGRESS** can create much smaller log files than when this variable is set to **ON** or **LOW**.

Observations in the field show that a high EMS log level (especially with the **-tm** option) not only creates large files, but also degrades performance more than expected.

K. Teamcenter core data dictionary

The following tables show the mandatory attributes for the core Teamcenter classes (that is, those attributes that are to be present for each class instance in the TC XML file).

> Note:
>
> When using the bulk load utility (**tcxml_import**), the utility sets appropriate values for attributes that are not mandatory. This type of activity must be done for other classes of interest as part of the customization work.

Item attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	
archive_info	No	
backup_date	No	
configuration_object_tag	No	
creation_date	No	
date_released	No	
ead_paragraph	No	
elemId	Yes	
global_alt_list	No	
gov_classification	No	
has_variant_module	No	
ip_classification	No	
is_configuration_item	No	
is_vi	No	
island_id	Yes	
item_id	Yes	
last_mod_date	No	

Attribute	Mandatory	Comments
last_mod_user	Yes	Points to the **User** element in the TC XML file.
license_list	No	
lsd	No	
object_application	Yes	Typically set to **Teamcenter**.
object_desc	No	
object_name	Yes	
object_properties	No	
object_type	Yes	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_organization	No	
owning_project	No	
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
parent_uid	Yes	
pid	No	**Class-id**. Fast import ignores this attribute.
preferred_global_alt	No	
process_stage_list	No	
project_list	No	
puid	Yes	
release_status_list	No	
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
timestamp	No	
uom_tag	No	
wso_thread	No	

Anchor elements

Attribute	Mandatory	Comments
acl_bits	No	
archive_date	No	
archive_info	No	
backup_date	No	
creation_date	No	
elemId	Yes	
immune_objects	No	
island_id	Yes	
keep_limit	Yes	Typically set to **3**.
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
lsd	No	
managed_objects	No	
object_properties	No	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	
puid	Yes	

ItemRevision attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	

Attribute	Mandatory	Comments
archive_info	No	
backup_date	No	
creation_date	No	
date_released	No	
declared_options	No	
ead_paragraph	No	
elemId	Yes	
gde_bvr_list	No	
gov_classification	No	
has_variant_module	No	
ip_classification	No	
island_id	Yes	
item_revision_id	Yes	
items_tag	Yes	Holds the **PUID** value of the corresponding Item.
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
license_list	No	
lsd	No	
object_application	No	
object_desc	No	
object_name	Yes	
object_properties	No	
object_type	Yes	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_organization	No	
owning_project	No	

Attribute	Mandatory	Comments
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
parent_uid	Yes	
pid	Yes	The **class-id**. Fast import ignores the attribute.
process_stage_list	No	
project_list	No	
puid	Yes	
release_status_list	No	
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
sequence_anchor	Yes	Holds the **PUID** value of the corresponding **Anchor** element.
sequence_id	Yes	Typically set to **1**.
sequence_limit	Yes	Typically set to **3**.
timestamp	No	
used_options	No	
variant_expression_block	No	
wso_thread	No	

Form attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	
archive_info	No	
backup_date	No	

Attribute	Mandatory	Comments
creation_date	No	
data_file	No	
date_released	No	
ead_paragraph	No	
elemId	Yes	
form_file	Yes	Typically set to **n/a**.
gov_classification	No	
ip_classification	No	
island_id	Yes	
last_mod_date	No	
last_mod_user		Points to the **User** element in the TC XML file.
license_list	No	
lsd	No	
object_application	Yes	Typically set to **Teamcenter**.
object_desc	No	
object_name	Yes	
object_properties	No	
object_type	Yes	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_organization	No	
owning_project	No	
owning_user	Yes	Points to the **User** element in the TC XML file.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	
pid	Yes	The **class-id**. Fast import ignores the attribute.

Attribute	Mandatory	Comments
process_stage_list	No	
project_list	No	
puid	Yes	
release_status_list	No	
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
timestamp	No	
wso_thread	No	

ImanRelation attributes

Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
lsd	No	
object_properties	No	
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	
pid	Yes	The **class-id** attribute. Fast Import ignores the attribute.
primary_object	Yes	Holds the **PUID** value of the primary object of the **ImanRelation** object.
puid	Yes	
relation_type	Yes	Points to the corresponding **ImanType** element in the TC XML file.
secondary_object	Yes	Holds the **PUID** value of the secondary object of the **ImanRelation** object.

Attribute	Mandatory	Comments
timestamp	No	
user_data	No	

PSBOMView attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	
archive_info	No	
backup_date	No	
creation_date	No	
date_released	No	
ead_paragraph	No	
elemId	Yes	
gov_classification	No	
ip_classification	No	
island_id	Yes	
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
license_list	No	
lsd	No	
object_application	Yes	Typically set to **Teamcenter**.
object_desc	No	
object_name	Yes	
object_properties	No	
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
object_type	Yes	

Attribute	Mandatory	Comments
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
parent_item	Yes	
parent_uid	Yes	
pid	No	The **class-id** attribute. Fast Import ignores the attribute.
process_stage_list	No	
project_list	No	
puid	Yes	
release_status_list	No	
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
timestamp	No	
view_type	Yes	Points to the **PSViewType** element in the TC XML file.
wso_thread	No	

PSBOMViewRevision attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	
archive_info	No	
backup_date	No	
bom_view	Yes	Points the **PUID** value of the corresponding **PSBOMView** element.
creation_date	No	
date_released	No	

Attribute	Mandatory	Comments
ead_paragraph	No	
elemId	Yes	
gov_classification	No	
island_id	Yes	
is_precise	Yes	Typically set to **0**.
ip_classification	No	
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
license_list	No	
lsd	No	
object_application	Yes	Typically set to **Teamcenter**.
object_desc	No	
object_name	Yes	
object_properties	No	
object_type	Yes	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_organization	No	
owning_project	No	
owning_site	Yes	Required for LL TC XML. Points to the **POM_imc** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
parent_uid	Yes	
pid	Yes	
process_stage_list	No	
project_list	No	
puid	Yes	

Attribute	Mandatory	Comments
release_status_list	No	
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
struct_last_mod_date	Yes	
timestamp	No	
wso_thread	No	

PSOccurrence attributes

Attribute	Mandatory	Comments
alternate_etc_ref	No	
child_item	Yes	Points the **PUID** value of the corresponding item.
child_bv	No	
effectivities	No	
elemId	Yes	
island_id	Yes	
lsd	No	
notes_ref	No	
object_properties	No	
occ_thread	Yes	Points the **PUID** value of the corresponding **PSOccurrenceThread** element in the TC XML file.
occ_flags	No	
occ_type	No	
occurrence_type	No	
order_no	No	Typically set to **10**.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_bvr	Yes	Points the **PUID** value of the corresponding

Attribute	Mandatory	Comments
		PSBOMViewRevision element in the TC XML file.
parent_uid	Yes	Required for low-level TC XML schema.
pid	No	
pred_list	No	
puid	Yes	
qty_value	No	Typically set to **-1**.
seq_no	Yes	Typically set to **10**.
timestamp	No	
uom_tag	No	
used_options	No	
variant_condition	No	
xform	No	

PSOccurenceThread attributes

Attribute	Mandatory	Comments
clone_stable_occ_uid	No	
elemId	Yes	
island_id	Yes	
lsd	No	
object_properties	No	
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	
pid	Yes	The **class-id** attribute. Fast import ignores this attribute.
puid	Yes	
timestamp	No	

Dataset attributes

Attribute	Mandatory	Comments
acl_bits	No	
active_seq	No	
archive_date	No	
archive_info	No	
backup_date	No	
creation_date	No	
dataset_type	Yes	Points to the **DatasetType** element in the TC XML file.
date_released	No	
ead_paragraph	No	
elemId	Yes	
format_used	No	
gov_classification	No	
ip_classification	No	
island_id	Yes	
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
license_list	No	
local_path	No	
lsd	No	
markup_acl	No	
markup_create_tool	No	
markup_official	No	
markup_status	No	
object_application	Yes	Typically set to **Teamcenter**.
object_desc	No	
object_name	Yes	

Attribute	Mandatory	Comments
object_properties	No	
object_type	Yes	
owning_organization	No	
owning_project	No	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
parent_uid	Yes	
pid	No	
process_stage_list	No	
project_list	No	
puid	Yes	
ref_names	No	
ref_list	Yes	Holds the **PUID** value of the corresponding **ImanFile** element in the TC XML file.
ref_types	No	
release_status_list	No	
rev_chain_anchor	Yes	Holds the **PUID** value of the corresponding **RevisionAnchor** element in the TC XML file.
revision_limit	Yes	Typically set to **1**.
revision_number	Yes	Typically set to **0**.
system_managed	No	
tool_used	Yes	Points to the **Tool** element in the TC XML file.
timestamp	No	

Attribute	Mandatory	Comments
user_class	No	
wso_thread	No	

RevisionAnchor attributes

Attribute	Mandatory	Comments
acl_bits	No	
archive_date	No	
archive_info	No	
backup_date	No	
creation_date	No	
elemId	Yes	
highest_rev	Yes	Typically set to **1**.
id	No	
island_id	Yes	
keep_limit	Yes	Typically set to **3**.
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
lsd	No	
object_properties	No	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	
pid	Yes	
puid	Yes	
rev	No	

Attribute	Mandatory	Comments
revisions	Yes	Holds the values of the corresponding **Dataset** revisions.
timestamp	No	

ImanFile attributes

Attribute	Mandatory	Comments
acl_bits	No	
archive_date	No	
archive_info	No	
backup_date	No	
creation_date	No	
destination_volume_tag	No	
elemId	Yes	
file_name	Yes	
hsm_info	No	
island_id	Yes	
last_mod_date	No	
last_mod_user	Yes	Points to the **User** element in the TC XML file.
lsd	No	
machine_type	No	
object_properties	No	
original_file_name	No	
owning_group	Yes	Points to the **Group** element in the TC XML file.
owning_user	Yes	Points to the **User** element in the TC XML file.
owning_site	Yes	Points to the **POM_imc** element in the TC XML file.
parent_uid	Yes	

Attribute	Mandatory	Comments
pid	Yes	The **class-id** attribute. Fast import ignores this attribute.
puid	Yes	
relative_directory_path	No	
released_version	No	
sd_path_name	Yes	Contains the relative path of the legacy file from the logical volume root.
status_flag	No	
store_and_forward_flag	No	
text_flag	No	
time_last_modified	No	
timestamp	No	
translate	No	
volume_tag	Yes	Points to the **ImanVolume** element in the TC XML file.
vm_info	No	

User attributes

Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
user_id	Yes	Candidate key

Group attributes

Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
name	Yes	Candidate key

Tool attributes

Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
object_name	Yes	Candidate key

ImanType attributes

Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
type_class	Yes	Candidate key
type_name	Yes	Candidate key

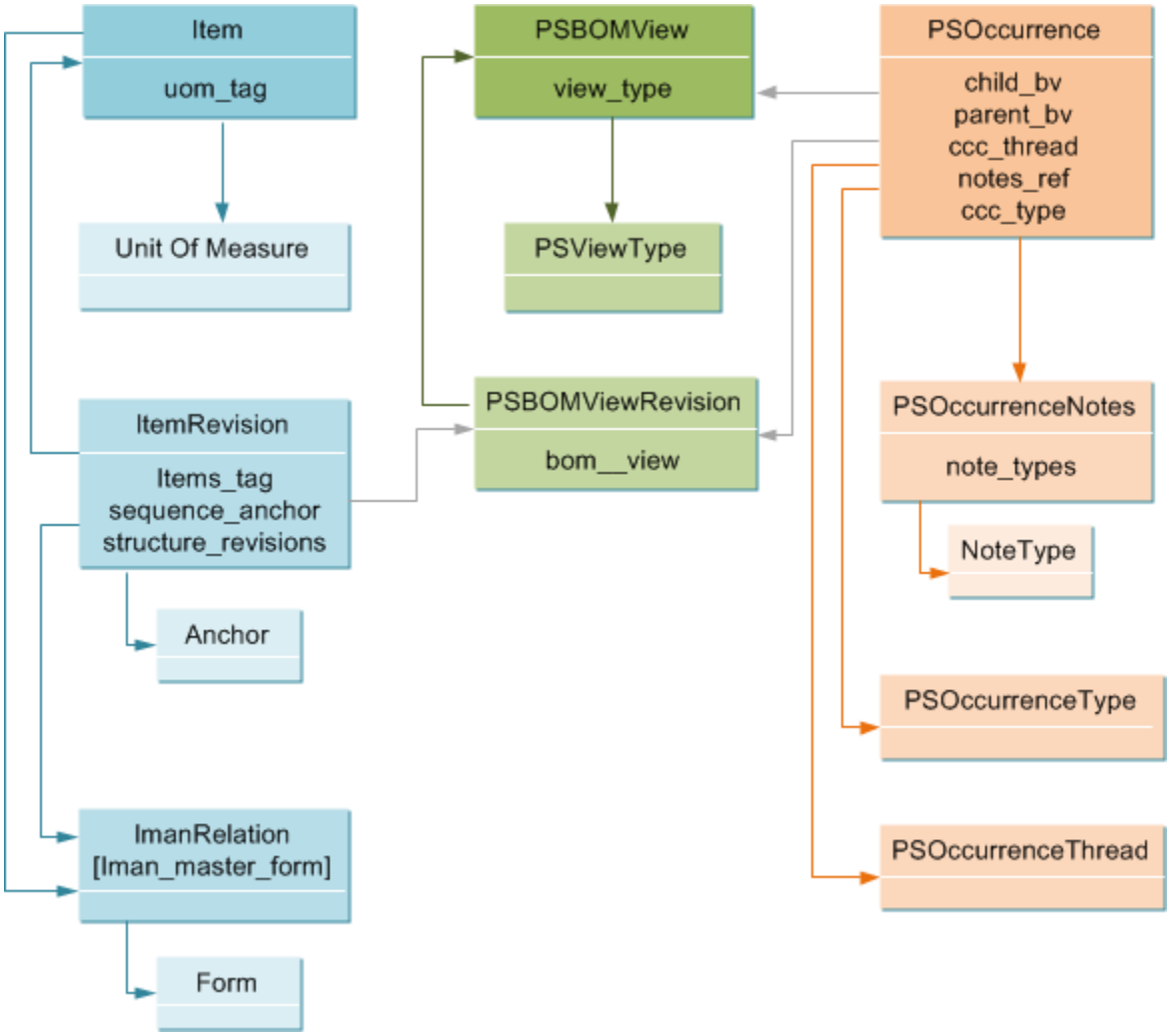
ImanVolume attributes

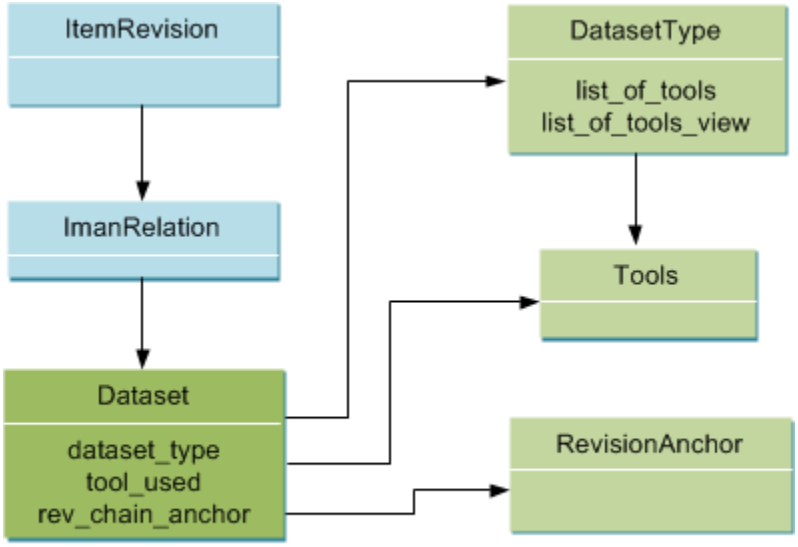
Attribute	Mandatory	Comments
elemId	Yes	
island_id	Yes	
volume_name	Yes	Candidate key

DatasetType attributes

Attribute	Mandatory	Comments
elemId	Yes	
datasettype_name	Yes	Candidate key
island_id	Yes	

L. Teamcenter data model diagram

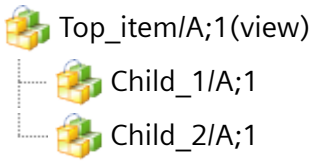




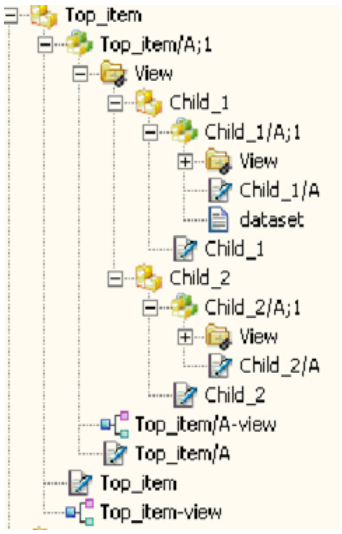
M. Sample assembly structure in a TC XML file

The structure in the following figure is a simple one-level bill of material (BOM) structure where:

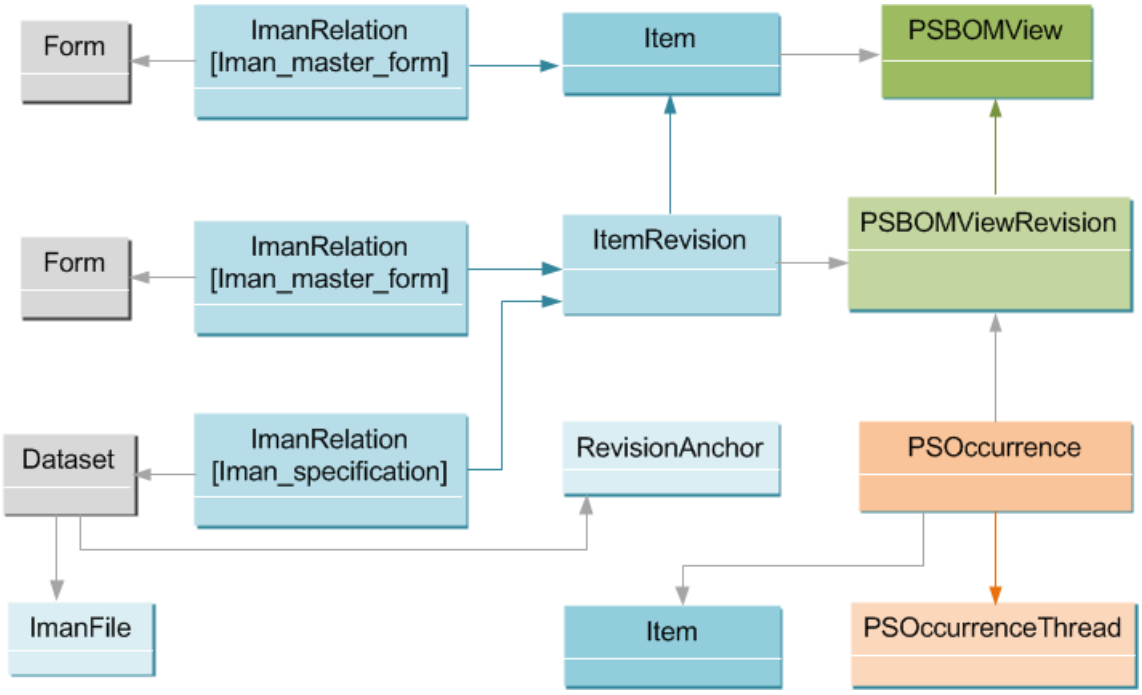
- The **Top_item** object is the parent item.
- The **Child_1** and **Child_2** objects are the children of the **Top_item** object.
- A dataset is attached to the **Child_1** object.



The expanded view of the **Top_item** object and its children includes the **dataset** object attached to the **Child_1** object:



The following figure shows the relationships between the objects.



N. Sample TC XML file with GSID references

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <TCXML xmlns="http://www.tcxml.org/Schemas/TCXMLSchema">
- <Group elemId="id1" island_id="0" name="dba">
  <GSIdentity elemId="id75" label="LegacyIDid75one" />
  </Group>
- <ImanType elemId="id2" island_id="0" type_class="ImanRelation"
type_name="IMAN_master_form">
  <GSIdentity elemId="id76" label="LegacyIDid76one" />
  </ImanType>
- <User elemId="id3" island_id="0" user_id="tcdba">
  <GSIdentity elemId="id77" label="LegacyIDid77one" />
  </User>
- <PSViewType elemId="id4" island_id="0" name="view">
  <GSIdentity elemId="id78" label="LegacyIDid78one" />
  </PSViewType>
+ <POM_imc elemId="id5" island_id="0" site_id="-1589622576">
+ <ItemRevision acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:07Z" date_released="" declared_options=""
ead_paragraph="" elemId="id6"
    gde_bvr_list="" gov_classification="" has_variant_module="" ip_classification=""
island_id="5"
   item_revision_id="A" items_tag="#id81" last_mod_date="2021-05-29T10:23:07Z"
last_mod_user="#id77"
   license_list="" lsd="2021-05-29T10:23:07Z" object_application="Teamcenter" object_desc=""
   object_name="onec" object_properties="0" object_type="ItemRevision" owning_group="#id75"
   owning_organization="" owning_project="" owning_user="#id77" parent_uid="#id81" pid="758"
   process_stage_list="" project_list="" release_status_list="" revision_limit="1"
revision_number="0"
   sequence_anchor="#id82" sequence_id="1" sequence_limit="3" timestamp="QpHx6UvxI45M1A"
used_options=""
   variant_expression_block="" wso_thread="">
- <Item acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
configuration_object_tag=""
   creation_date="2021-05-29T10:23:06Z" date_released="" ead_paragraph="" elemId="id7"
global_alt_list=""
   gov_classification="" has_variant_module="" ip_classification="" is_configuration_item=""
is_vi=""
   island_id="5" item_id="onec" last_mod_date="2021-05-29T10:23:07Z" last_mod_user="#id77"
license_list=""
   lsd="2021-05-29T10:23:07Z" object_application="Teamcenter" object_desc=""
object_name="onec"
   object_properties="0" object_type="Item" owning_group="#id75" owning_organization=""
owning_project=""
   owning_user="#id77" parent_uid="" pid="564" preferred_global_alt="" process_stage_list=""
project_list=""
   release_status_list="" revision_limit="1" revision_number="0" timestamp="QpIx6UvxI45M1A"
uom_tag=""
   wso_thread="">
   <GSIdentity elemId="id81" label="LegacyIDid81one" />
  </Item>
- <Anchor acl_bits="0" archive_date="" archive_info="" backup_date=""
creation_date="2021-05-29T10:23:07Z"
   elemId="id8" immune_objects="" island_id="5" keep_limit="3"
last_mod_date="2021-05-29T10:23:07Z"
   last_mod_user="#id77" lsd="2021-05-29T10:23:07Z" managed_objects="" object_properties="0"
   owning_group="#id75" owning_user="#id77" parent_uid="#id80" pid="520"
```

```
timestamp="QlMx6UvxI45M1A">
   <GSIdentity elemId="id82" label="LegacyIDid82one" />
  </Anchor>
- <ImanRelation elemId="id9" island_id="5" lsd="2021-05-29T10:23:06Z" object_properties="0"
parent_uid="#id81"
   pid="443" primary_object="#id81" relation_type="#id76" secondary_object="#id85"
timestamp="QhGx6UvxI45M1A"
   user_data="">
   <GSIdentity elemId="id83" label="LegacyIDid83one" />
  </ImanRelation>
- <ImanRelation elemId="id10" island_id="5" lsd="2021-05-29T10:23:07Z" object_properties="0"
parent_uid="#id80"
   pid="443" primary_object="#id80" relation_type="#id76" secondary_object="#id86"
timestamp="QpEx6UvxI45M1A"
   user_data="">
   <GSIdentity elemId="id84" label="LegacyIDid84one" />
  </ImanRelation>
- <Form acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:06Z" data_file="" date_released="" ead_paragraph=""
elemId="id11"
   form_file="n/a" gov_classification="" ip_classification="" island_id="5"
   last_mod_date="2021-05-29T10:23:06Z" last_mod_user="#id77" license_list=""
lsd="2021-05-29T10:23:06Z"
   object_application="Teamcenter" object_desc="" object_name="onec" object_properties="0"
   object_type="Item Master" owning_group="#id75" owning_organization="" owning_project=""
owning_user="#id77"
   parent_uid="#id83" pid="555" process_stage_list="" project_list="" release_status_list=""
   revision_limit="1" revision_number="0" timestamp="QdEx6UvxI45M1A" wso_thread="">
   <GSIdentity elemId="id85" label="LegacyIDid85one" />
  </Form>
- <Form acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:07Z" data_file="" date_released="" ead_paragraph=""
elemId="id12"
   form_file="n/a" gov_classification="" ip_classification="" island_id="5"
   last_mod_date="2021-05-29T10:23:07Z" last_mod_user="#id77" license_list=""
lsd="2021-05-29T10:23:07Z"
   object_application="Teamcenter" object_desc="" object_name="onec/A" object_properties="0"
   object_type="ItemRevision Master" owning_group="#id75" owning_organization=""
owning_project=""
   owning_user="#id77" parent_uid="#id84" pid="555" process_stage_list="" project_list=""
   release_status_list="" revision_limit="1" revision_number="0" timestamp="QpBx6UvxI45M1A"
wso_thread="">
  <GSIdentity elemId="id86" label="LegacyIDid86one" />
  </Form>
- <ItemRevision acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:35Z" date_released="" declared_options=""
ead_paragraph="" elemId="id13"
   gde_bvr_list="" gov_classification="" has_variant_module="" ip_classification=""
island_id="3"
   item_revision_id="A" items_tag="#id88" last_mod_date="2021-05-29T10:24:26Z"
last_mod_user="#id77"
   license_list="" lsd="2021-05-29T10:24:26Z" object_application="Teamcenter" object_desc=""
   object_name="one" object_properties="0" object_type="ItemRevision" owning_group="#id75"
   owning_organization="" owning_project="" owning_user="#id77" parent_uid="#id88" pid="758"
   process_stage_list="" project_list="" release_status_list="" revision_limit="1"
revision_number="0"
   sequence_anchor="#id93" sequence_id="1" sequence_limit="3" structure_revisions="#id92"
   timestamp="gBLx6UvxI45M1A" used_options="" variant_expression_block="" wso_thread="">
   <GSIdentity elemId="id87" label="LegacyIDid87one" />
  </ItemRevision>
```

```
- <Item acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
bom_view_tags="#id90"
   configuration_object_tag="" creation_date="2021-05-29T10:23:35Z" date_released=""
ead_paragraph=""
   elemId="id14" global_alt_list="" gov_classification="" has_variant_module=""
ip_classification=""
   is_configuration_item="" is_vi="" island_id="3" item_id="one"
last_mod_date="2021-05-29T10:24:26Z"
   last_mod_user="#id77" license_list="" lsd="2021-05-29T10:24:26Z"
object_application="Teamcenter"
   object_desc="" object_name="one" object_properties="0" object_type="Item"
owning_group="#id75"
   owning_organization="" owning_project="" owning_user="#id77" parent_uid="" pid="564"
   preferred_global_alt="" process_stage_list="" project_list="" release_status_list=""
revision_limit="1"
   revision_number="0" timestamp="gBMx6UvxI45M1A" uom_tag="" wso_thread="">
   <GSIdentity elemId="id88" label="LegacyIDid88one" />
  </Item>
- <PSOccurrence alternate_etc_ref="" child_bv="" child_item="#id81" effectivities=""
elemId="id15"
   island_id="3" lsd="2021-05-29T10:24:26Z" notes_ref="" object_properties="0" occ_flags="0"
   occ_thread="#id91" occ_type="" occurrence_type="0" order_no="10" parent_bvr="#id92"
parent_uid="#id92"
   pid="783" pred_list="" qty_value="-1" seq_no="10" timestamp="gBEx6UvxI45M1A" uom_tag=""
used_options=""
   variant_condition="" xform="">
   <GSIdentity elemId="id89" label="LegacyIDid89one" />
  </PSOccurrence>
- <PSBOMView acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:24:16Z" date_released="" ead_paragraph="" elemId="id16"
   gov_classification="" ip_classification="" island_id="3"
last_mod_date="2021-05-29T10:24:16Z"
   last_mod_user="#id77" license_list="" lsd="2021-05-29T10:24:16Z"
object_application="Teamcenter"
   object_desc="" object_name="one-View" object_properties="0" object_type="BOMView"
owning_group="#id75"
   owning_organization="" owning_project="" owning_user="#id77" parent_item="#id88"
parent_uid="#id88"
   pid="594" process_stage_list="" project_list="" release_status_list="" revision_limit="1"
   revision_number="0" timestamp="Q5Nx6UvxI45M1A" view_type="#id78" wso_thread="">
   <GSIdentity elemId="id90" label="LegacyIDid90one" />
  </PSBOMView>
- <PSOccurrenceThread clone_stable_occ_uid="AAAAAAAAAAAAAA" elemId="id17" island_id="3"
   lsd="2021-05-29T10:24:26Z" object_properties="0" parent_uid="#id89" pid="256"
timestamp="gBDx6UvxI45M1A">
   <GSIdentity elemId="id91" label="LegacyIDid91one" />
  </PSOccurrenceThread>
- <PSBOMViewRevision acl_bits="0" active_seq="1" archive_date="" archive_info=""
backup_date=""
   bom_view="#id90" creation_date="2021-05-29T10:24:17Z" date_released="" ead_paragraph=""
elemId="id18"
   gov_classification="" ip_classification="" is_precise="0" island_id="3"
   last_mod_date="2021-05-29T10:24:26Z" last_mod_user="#id77" license_list=""
lsd="2021-05-29T10:24:26Z"
   object_application="Teamcenter" object_desc="" object_name="one/A-View"
object_properties="0"
   object_type="BOMView Revision" owning_group="#id75" owning_organization=""
owning_project=""
   owning_user="#id77" parent_uid="#id87" pid="595" process_stage_list="" project_list=""
   release_status_list="" revision_limit="1" revision_number="0"
```

```
struct_last_mod_date="2021-05-29T10:24:26Z"
   timestamp="gBCx6UvxI45M1A" wso_thread="">
   <GSIdentity elemId="id92" label="LegacyIDid92one" />
  </PSBOMViewRevision>
- <Anchor acl_bits="0" archive_date="" archive_info="" backup_date=""
creation_date="2021-05-29T10:23:35Z"
   elemId="id19" immune_objects="" island_id="3" keep_limit="3"
last_mod_date="2021-05-29T10:23:35Z"
   last_mod_user="#id77" lsd="2021-05-29T10:23:35Z" managed_objects="" object_properties="0"
   owning_group="#id75" owning_user="#id77" parent_uid="#id87" pid="520"
timestamp="QtIx6UvxI45M1A">
   <GSIdentity elemId="id93" label="LegacyIDid93one" />
  </Anchor>
- <ImanRelation elemId="id20" island_id="3" lsd="2021-05-29T10:23:35Z" object_properties="0"
parent_uid="#id88"
   pid="443" primary_object="#id88" relation_type="#id76" secondary_object="#id96"
timestamp="QtCx6UvxI45M1A"
   user_data="">
   <GSIdentity elemId="id94" label="LegacyIDid94one" />
  </ImanRelation>
- <ImanRelation elemId="id21" island_id="3" lsd="2021-05-29T10:23:35Z" object_properties="0"
parent_uid="#id87"
   pid="443" primary_object="#id87" relation_type="#id76" secondary_object="#id97"
timestamp="QtPx6UvxI45M1A"
   user_data="">
   <GSIdentity elemId="id95" label="LegacyIDid95one" />
  </ImanRelation>
- <Form acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:35Z" data_file="" date_released="" ead_paragraph=""
elemId="id22"
   form_file="n/a" gov_classification="" ip_classification="" island_id="3"
   last_mod_date="2021-05-29T10:23:35Z" last_mod_user="#id77" license_list=""
lsd="2021-05-29T10:23:35Z"
   object_application="Teamcenter" object_desc="" object_name="one" object_properties="0"
   object_type="Item Master" owning_group="#id75" owning_organization="" owning_project=""
owning_user="#id77"
   parent_uid="#id94" pid="555" process_stage_list="" project_list="" release_status_list=""
   revision_limit="1" revision_number="0" timestamp="QpPx6UvxI45M1A" wso_thread="">
   <GSIdentity elemId="id96" label="LegacyIDid96one" />
  </Form>
- <Form acl_bits="0" active_seq="1" archive_date="" archive_info="" backup_date=""
   creation_date="2021-05-29T10:23:35Z" data_file="" date_released="" ead_paragraph=""
elemId="id23"
   form_file="n/a" gov_classification="" ip_classification="" island_id="3"
   last_mod_date="2021-05-29T10:23:35Z" last_mod_user="#id77" license_list=""
lsd="2021-05-29T10:23:35Z"
   object_application="Teamcenter" object_desc="" object_name="one/A" object_properties="0"
   object_type="ItemRevision Master" owning_group="#id75" owning_organization=""
owning_project=""
   owning_user="#id77" parent_uid="#id95" pid="555" process_stage_list="" project_list=""
   release_status_list="" revision_limit="1" revision_number="0" timestamp="QtMx6UvxI45M1A"
wso_thread="">
   <GSIdentity elemId="id97" label="LegacyIDid97one" />
  </Form>
- <Header author="tcdba" date="2021-05-29" elemId="id24" originatingSite="-2132200621"
targetSite=""
   time="15:58:52">
- <TransferFormula elemId="id25">
- <OptionSet elemId="id26" name="SiteConsolidationDefault">
  <Option elemId="id27" name="internalClosureRule" value="True" />
```

```
  <Option elemId="id28" name="opt_entire_bom" value="True" />
  <Option elemId="id29" name="opt_entire_mse" value="False" />
  <Option elemId="id30" name="opt_ixr_islanchor" value="False" />
  <Option elemId="id31" name="opt_mechatro" value="False" />
  <Option elemId="id32" name="opt_res_audit" value="True" />
  <Option elemId="id33" name="opt_res_checkout" value="False" />
  <Option elemId="id34" name="opt_varexp_islanchor" value="True" />
  </OptionSet>
- <SessionOptions elemId="id35">
  <Option elemId="id36" name="fastStream" value="yes" />
  </SessionOptions>
  <TransferMode elemId="id37" object_name="SiteConsolidationDefaultTM" />
  <Reason elemId="id38" />
  </TransferFormula>
  </Header>
  </TCXML>
```

O. Frequently asked questions about bulk data loading

Consideration	Description

Why is there a different referencing scheme used within the TC XML format?	To expand on this question with examples, for **owning_user** the element id is used, for the referencing **items_tag**, the **uid** of the parent id is used. Also the **parent_uid** attribute is mentioned twice as **items_tag** and **parent_uid**. To be able to separate the users/objects/relations in the loads it would be best to reference everything by the Teamcenter UID.
	
	`<ItemRevision acl_bits="0" active_seq="1" archive_date=""`
	`archive_info="" backup_date=""`
	`creation_date="2010-06-29T15:43:21Z" date_released=""`
	`declared_options="" ead_paragraph=""`
	`elemId="id17" gde_bvr_list="" gov_classification=""`
	`has_variant_module=""`
	`ip_classification="" island_id="4" item_revision_id="A"`
	`items_tag="AVDxiag7I45M1A"`
	`last_mod_date="2010-06-29T15:46:31Z" last_mod_user="#id5"`
	`license_list=""`
	`lsd="2010-06-29T15:46:31Z" object_application="Teamcenter"`
	`object_desc="" object_name="child1"`
	`object_properties="0" object_type="ItemRevision"`
	`owning_group="#id1" owning_organization=""`
	`owning_project="" owning_user="#id5"`
	`parent_uid="AVDxiag7I45M1A" pid="758"`
	`process_stage_list="" project_list="" puid="AVMxiag7I45M1A"`
	`release_status_list=""`
	`revision_limit="1" revision_number="0"`
	`sequence_anchor="AVOxiag7I45M1A" sequence_id="1"`
	`sequence_limit="3" structure_revisions="QFJxiag7I45M1A"`
	`timestamp="QNMxiag7I45M1A"`
	`used_options="" variant_expression_block="" wso_thread=""/>`
	
	Organization and administrative objects, such as **User** and **Group**, cannot be referenced by a **UID**. They have to be looked up based on their candidate key. So there is an XML element for the organization object with just the candidate key and element ID and all the references use the **elemId** attribute to point to it.

How are the island IDs handled?	The import process collects all the objects that have the same **island-id** attribute and saves them at once.
Why not write header information to a separate configuration file?	The import process relies on originating site information and the session options in the **Header** element. This information is read in the first pass of the XML parsing and used in the second pass when the actual import happens.

Consideration	Description
Which attributes in the TC XML are mandatory? Can empty attributes can be left out?	The fast import sets null values for missing attributes that do not have an initial value. However, there is no straightforward way to arrive at a set of attributes that are required to be populated for a given class. The **tcxml_validate** utility reports if the required typed and untyped references are missing from the TC XML. In addition, the list of mandatory attributes for core Teamcenter classes documented in *Teamcenter core data dictionary*. For other classes of interest, similar exercise must be done as part of the customization work.
How is the time stamp generated?	The time stamp shows when the object was last modified. The **timestamp** attribute takes the form of a UID – not a date or time as might be expected. Because a UID encodes a representation of time, a UID is sufficient for this purpose. This attribute can be omitted from the TC XML. Fast import internally generates and sets the time stamp for an object.
What are the restrictions for the external and site ID values used with **TIE_get_hashed_uid**?	When generating the UID with **TIE_get_hashed_uid**, an external ID coming from the legacy system and a site ID identifying the legacy system are required. These ID values have no character set or string length restrictions.
Is the hashed UID the same as the external ID used for creating the UID internally with the bulk loader process? Or what is the format of the GSID needed?	The GSID contains **label**, **sublabel**, and **split_token** attributes to allow for many-to-many mappings. If the external ID is composed such that it is unique across the legacy data model it can be used to populate the **label** attribute.
Must the **elemId** and **island_id** attributes be unique in a file or unique over all data loaded?	The **elemId** and **island_id** attributes are unique per file.
What causes errors during job scheduling and execution?	Process execution problems may occur due to available memory or maximum time-to-live (ttl) expiration when running in the Apache ODE in memory. This issue is usually encountered when you are running the ODE on a JBoss application server and the ODE uses an Oracle database. To run an ODE process in memory, uncomment or add the **<in-memory>true</in-memory>** element to the *ode-working-dir***/processes/data-transfer/deploy.xml** file. For more information about ODE processes, see **https://ode.apache.org**. If your processes may exist for longer than 10 minutes, configure the in-memory ttl to longer than 10 minutes using the **ode-axis2.mex.inmem.ttl** property. This properties value is set in milliseconds in the *ode-working-dir***/conf/ode-axis2.properties** file. For example, the following set the ttl to 30 minutes:

Consideration	Description
	`ode-axis2.mex.inmem.ttl=1800000`