TEAMCENTER

Product Data Validation

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

Validation rules

1. Getting started with Validation Manager

What is Validation Manager?

The Validation Manager application in Teamcenter provides a decision mechanism to let users ensure a target meets compliance before it is released. Once validation processes are configured, you can select a target in Teamcenter and confirm its compliance with the business rules defined at your site.

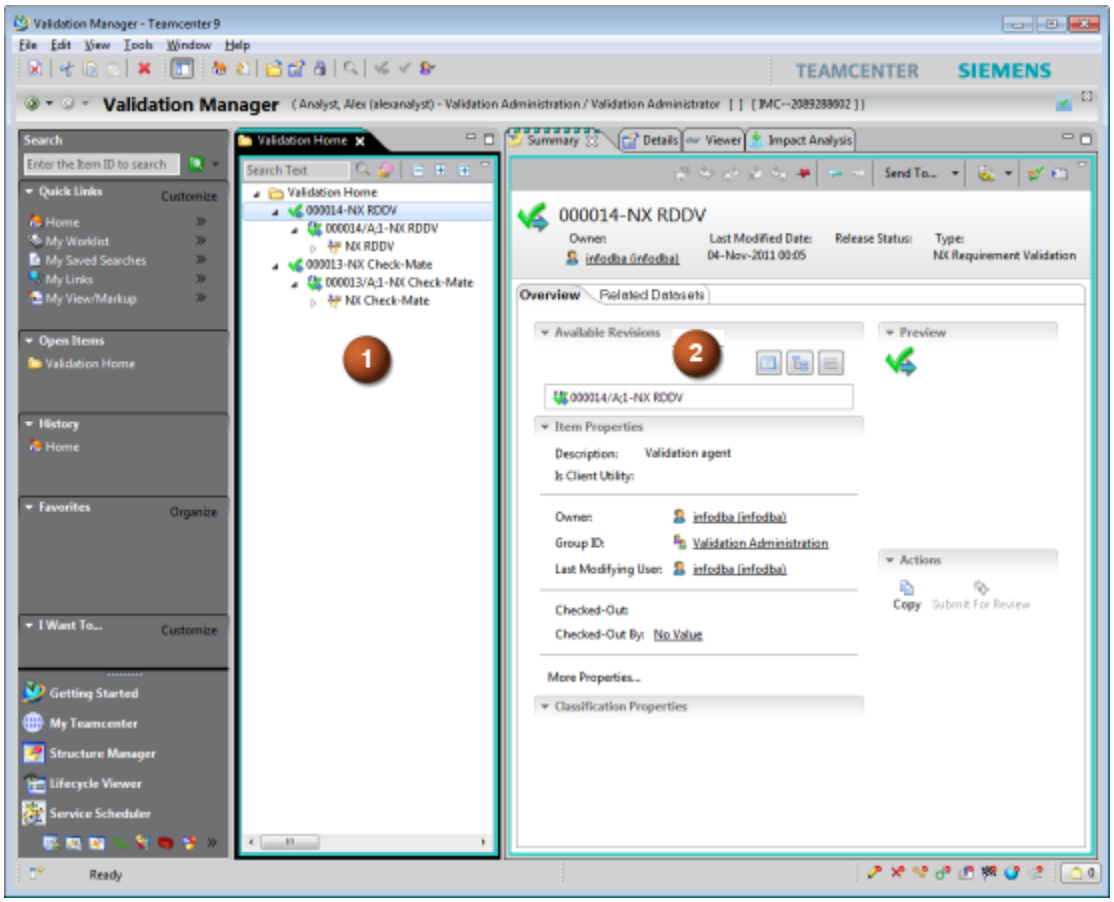
The validation objects in Teamcenter are validation agents with their checkers and validation results. In Validation Manager, you define or configure validation agents that contain the information of a particular validation software application, such as NX Check-Mate and NX RDDV (Requirement-Driven Design Validation). Each agent has executable checkers, which define the parameters that the agent checks. When you run the checkers, they validate the item revisions and store the data in validation result objects, where you can view it.

Requirements for using Validation Manager

Prerequisites	You do not need any special permissions to view validation agents, checkers, and results in the Validation Manager application. You must have Validation Manager administrator or Teamcenter administrator privileges to create, modify, and delete validation agents and checkers in the Validation Manager.
Enable Validation Manager	Validation Manager does not need to be enabled before you use it.
Configure Validation Manager	The Validation Manager administrator or the system administrator must **configure one or more validation agents and validation checkers**. The Validation Manager administrator or the system administrator may also need to modify the preferences settings for Validation Manager.
Start Validation Manager	Click **Validation Manager** in the navigation pane.

Validation Manager interface

Validation Manager interface overview



1	**Validation Agent** view	Displays the **Validation Home** tree pane.
2	**Summary** view	For the **Validation Agent** view, displays details about the selected agent or checker.

Validation Manager menu commands

File→New menu commands

Command	Description
Validation Agent	Displays the **New Agent** dialog box in which you choose an agent type and specify the required properties for it. The properties shown in this dialog box are defined by the Business

Command	Description
	Modeler IDE and vary depending on the agent type you choose.
Validation Checker	Displays the **New Checker** dialog box in which you choose a checker type and specify the required properties for it. The properties shown in this dialog box are defined by the Business Modeler IDE and vary depending on the checker type you choose.

You must have Validation Manager administrator or system administrator privileges to create new agent or checker objects.

Edit menu commands

Command	Description
Cut	Removes a selected data object reference from the current location and places it on the clipboard. You must have read privileges on the object and write privileges on its container to move or remove an object.
Copy	Replicates a data object reference in another application. You must have read privileges for the selected object that you want to copy. You can also create a copy by dragging the object to another Teamcenter application.
Paste	Moves a data object reference from the clipboard to the current location in the data object area. It is important to select the proper destination for the data object reference before choosing the **Paste** menu command. You must have read and write privileges to the destination object. If your administrator has defined mandatory properties for the type of relation by which the pasted object reference (secondary object) is associated to the primary object, a dialog box lets you define attribute values for the objects.
Delete	Deletes the selected object.
User Setting	Provides access to the **User Setting** dialog box used to: • View, define, or change company and group profile information for a user. • View, define, or change settings for a group, role, and/or volume. • View or change logging privileges for a user with **dba** privileges.

Command	Description
Options	Lets you set preferences for a wide variety of user interface and application display and processing attributes.
Take Ownership	Transfers ownership of selected agent or checker to the currently logged-on Validation Manager administrator.

Validation Manager buttons

Button	Description
Soft Abort 	Stops the process. This button is active when an application is loading or when the system is processing data for a task. However, the soft abort operation is a logical interrupt that can be performed only when the system encounters an interruption between two processes. The following examples illustrate some situations in which you can use the **Soft Abort** button: • Creating a folder When a folder is created, the system creates the folder, pastes the folder, and opens the folder (if **Open on Create** is selected). The only points in this process at which the **Soft Abort** button can be used to stop the process are between when the folder has been created and is about to be pasted or when the folder has been pasted and is about to be opened. • Deleting objects When objects are deleted, the system loads the objects and then deletes them. The only point in this process at which the **Soft Abort** button can be used to stop the process is between when the object is loaded and when it is deleted, or if deleting multiple objects, the process can be stopped between when the previous object has been deleted and the next object is loaded. • Running searches When a query is run, the system executes the query and then loads the objects. The only point at which the operation can be stopped is after the query has run but before the objects are loaded.
Cut 	Removes a selected data object reference and places it on the clipboard. You must have read privileges on the object and write

Button	Description
	privileges on its container (object folder, item revision) to cut an object reference.
Copy	Duplicates an object reference. You must have read privileges for the object that you want to copy. Tip: You can create a copy of an object reference in a different application by dragging the object from My Teamcenter to the other application.
Paste	Pastes the contents of the clipboard into the selected container object.
Delete	Deletes the selected object.
Open home folder	Opens your **Home** folder view in the **Validation Manager** perspective.
Refresh	Refreshes the display in your rich client window.
Open the selected object	Opens a selected data object—folder, item, item revision, or dataset—to access product information.
Properties	Displays the properties of a selected object. You can also use this option to edit object properties.
Access	Allows you to select a user and view, change, or apply access permissions for a selected object.
Search	Displays the **Search** pane used to execute database queries.
Agent	Displays the **New Agent** dialog box in which you choose an agent type and specify the required properties for it. The properties shown in this dialog box are defined by Business Modeler IDE and vary depending on the agent type you choose.
Checker	Displays the **New Checker** dialog box in which you choose a checker type and specify the required properties for it. The properties shown in this dialog box are defined by Business Modeler IDE and vary depending on the checker type you choose.
Take Ownership	Transfers ownership of the selected agent or checker to the currently logged-on Validation Manager administrator.

Validation Manager commands in My Teamcenter

You use My Teamcenter menu commands to validate targets and view results.

Command	Description
Run Validations	Displays the **Run Validation** dialog box listing available validation agents and checkers and selected validation agents and checkers.
View Results	Displays **Validation Results Viewer** in the Validation Manager perspective to show the validation results of the selected targets. Results are rendered with these icons: **Show Not-Run**, **Show Out-of-Date**, **Show Up-to-Date**, **Show Passed**, **Show Failed**, **Show Passed Evaluation**, and **Show Failed evaluation**.

What are perspectives and views?

Within the rich client user interface, application functionality is provided in *perspectives* and *views*.

View	The basic display component that displays related information in a UI window.
Perspective	A collection of one or more views and their layout.

Some applications use a perspective with multiple views to arrange how functionality is presented. Other applications use a perspective with a single view.

You can use the **HiddenPerspectives** preference to prevent the display of some Teamcenter perspectives in the rich client.

If your site has online help installed, you can access the application and view help from the rich client **Help** menu.

Basic concepts for using Validation Manager

Validation agents and checkers are used to integrate a validation application to ensure target objects meet a certain type of compliance. Validation Manager lets you select an agent and checkers to start the validation process on the selected target objects.

- A validation agent defines how to perform a validation on the target objects.

  A validation agent can have one or multiple validation checkers, depending on the logic in each integrated validation application.

- A validation checker may provide the information on a specific validation functionality that a validation application performs.

A validation checker is created in Teamcenter under its associated validation agent.

Teamcenter already provides agent and checker classes to integrate with these validation applications:

- NX Check-Mate
- NX RDDV (Requirement-Driven Design Validation)
- BOM Grading

NX Check-Mate and NX RDDV agents are created when the database is initially populated. You can create checkers for these validation applications and agents and checkers for others in Validation Manager.

To create a custom validation agent to integrate with a custom validation application, you must run the Business Modeler IDE application to create new business objects that are used to instantiate the custom validation agent and validation data objects. You can then **create agents and checkers** for the custom application in Validation Manager.

Validation Manager lets you **specify validation rules** to verify the selected targets. Validation rules are defined in a validation rule XML file. The validation rule XML file is stored in a **ValidationRuleSet** dataset. The **ValidationRuleSet** dataset is then used to examine validation results on a target to decide whether a target conforms to standards and can be released or baselined.

Validation results contain information on a validation target object. Validation Manager lets you view and manage the validation results. Validation results then can be used as the basis for deciding whether a target conforms to standards and can be released. Validation Manager stores the results of the validation process in a viewable form associated with the selected item revision.

Basic tasks using Validation Manager

Validation Manager administrator or Teamcenter administrator tasks

The Validation Manager administrator role is a role under the **Validation Administration** group that can perform validation administrative tasks.

- Prior to starting the validation process, a Validation Manager administrator must create some validation agents and validation checkers. The Validation Manager administrator can use the Validation Manager **File→New** menu commands to create new validation agents and checkers.
- A Validation Manager administrator can also modify or delete validation agents and validation checkers.

The **Override Approver** role is a role under the **Validation Administration** group that can approve or reject override requests. The Override Approver can approve or reject the override request using the **Validation Results Viewer** view or a Workflow process.

> Note:
>
> It is a Teamcenter administrative task to create new users with the **Validation Administration** group and roles or add existing users into **Validation Administration** group and roles before using Validation Manager.

There are various Teamcenter administrative tasks related to the Validation Manager or validation objects:

- Create new users with, or add existing users to, **Validation Administration** groups and roles.
- Define closure rules for traversal scope of a validation checker for NX Check-Mate.
- Configure the tool for the NX Check-Mate agent to use.
- Configure Validation Manager preferences to control behavior.
- Configure the workflow to approve the override request of validation results using these templates and handlers:
  - **ResultOverrideApproval**
  - **VAL-approve-result-overrides**
  - **VAL-reject-result-overrides**
  - **VAL-set-condition-result-overrides**
- Configure the workflow and baseline operation to evaluate validation results based on validation rules using these handlers:
  - **VAL-check-validation-result**
  - **VAL-check-validation-result-with-rules**
  - **VAL-set-condition-by-check-validation-result**
- Configure the PLM XML transfer mode for import/export validation results.
- Configure the Business Modeler IDE handler to copy validation results when the target object is copied.

Validation Manager user tasks

Users can perform the following typical tasks related to Validation Manager or validation objects.

- **Run validations**

  After a Validation Manager administrator defines some validation agents and validation checkers, any user can select some objects, and choose Teamcenter **Tools→Validation→Run Validation**, and then select some validation checkers under an agent to run validation against the selected objects. After the validation process completes, the corresponding validation results are generated by the agent.

- **View results**

  You can view the validation results associated with selected objects in the **Validation Results Summary** view or you can view the results with or without rule evaluation in the **Validation Results Viewer** view.

- **Override results**

  You can request to override a certain validation result status from fail to pass or vice versa by creating an override request and selecting an Override Approver to review the request using the **Validation Results Viewer**. You can also start a workflow process to assign the Override Approver to review the request

2. Setting up Validation Manager

Configuring validation processes

Validation object functionality is used as a decision mechanism, ensuring an item revision meets compliance before it is released. After validation processes are configured, users can select an item revision from My Teamcenter and confirm its compliance with the business rules defined at your site.

Before object validation functionality can be run at your site, a system administrator must configure an environment for all the validation applications intended to be used for data validation. The environment must be configured for all the validation applications on a server machine capable of supporting all the CAD applications.

The CAD applications used to validate the data must be configured to perform validations for the defined objects. You do this by creating validation data objects for each desired validation. To enable object validation functionality, validation data objects must be defined within the relevant CAD applications.

To enable object validation functionality, validation agents and checkers must be defined within the relevant applications.

For example, to perform three validation functionalities in the NX Check-Mate validation application and another three validation functionalities in another validation application, create two validation agents for these two validation applications and three validation checkers under each validation agent.

You must be a Validation Manager administrator or system administrator to create validation data objects.

Setting up Validation Manager

A Validation Manager administrator or Teamcenter administrator uses the Validation Manager application to create validation agents and checkers.

The applications used to validate the data must be configured to perform validations for the defined objects. You do this by creating validation agents and checkers for each desired application.

The administrator must create the following validation agents and checkers:

- One or more validation agents
- One or more validation checkers

Setting up the NX Check-Mate application

Configure the NX Check-Mate agent

The NX Check-Mate agent is added automatically when you install Teamcenter or upgrade your database. You need to configure the agent only if you want settings other than the defaults.

Read access is required for NX Integration to allow NX Check-Mate to retrieve and save results from and to the Teamcenter database.

1. Go to the Validation Manager application in Teamcenter and select **Validation Home**.

2. Expand the **NX Check-Mate** agent and agent revision.

3. (Optional) Modify the properties:

   - Set the **Is Client Utility** agent property to **False** to run the validation process on the Teamcenter server workstation. Set it to **True** to run the validation process on the rich client. The default is **False**.

   - Set the **Keep Overridden Result** agent revision property to **True** to prevent regeneration of all permanently overridden validation results of a validation agent when the tests are run. Set it to **False** to update all permanently overridden validation results of a validation agent when the tests are run. The default is **False**.

   - Set the **Override Reason Is Mandatory** agent revision property to **True** to require a reason when you request an override. You must also provide a category and override reason details when you fill in the **Create Override** dialog box. The default is **False**.

4. (Optional) Provide a **validation closure rule** used to select targets for the validation. The default is **NX Parts**.

5. (Optional) Provide utility arguments for the command string used to start the validation utility. For example, the **-pim** argument name, the **=** argument operation, and the **yes** argument value comprise a **-pim=yes** argument.

6. (Optional) Select a tool to attach to the agent. The default is set to the NX Check-Mate tool.

Create an NX Check-Mate checker

When creating checkers for use in Validation manager. Siemens Digital Industries Software recommends that you let NX initially configure the Check-Mate checkers with their correct class names, and then manually change the checker configuration if you want settings other than the defaults. If configured and you are using NX integration with Teamcenter, when run through the UI NX will create checkers automatically under the configured NX check mate agent the first time that profile is used to check a

part. The automatic saving will occur if the options to save the validation results back to Teamcenter are set.

Alternatively NX checkers will be created and automatically stored under the configured NX check mate agent if the command line utility ug_check_part is executed.

To identify the NX checker class names, see the NX DFA Check-Mate file.

1. Expand **Validation Home** and select the **NX Check-Mate** agent revision.

2. Choose **File→New→Validation Checker**.

3. Select **NX Check-Mate Checker** and click **Next**.

4. Fill in the checker **Name** and the **Checker Class Name** boxes. The **Name** box is the display name for the checker, while the **Checker Class Name** box is the internal checker class name used by the NX Check-Mate application.

5. (Optional) Provide a description to let users know what the checker verifies.

6. Click **Next**.

7. Specify whether the checker is a mandatory checker. If the checker is mandatory, when you choose **Tools→Validation→Run Validation**, the checker is marked with a red star at the end when displayed in the **Run Validations** dialog box. If the **TC_VALIDATION_mandatory_checkers_required** preference is set to **true**, after you select an agent revision, all the mandatory validation checkers under the agent are automatically added to the **Selected Validation Checkers** tree at the right side, and they can not be removed unless you select another agent revision.

8. (Optional) Provide a category name to let users know what type of information the checker verifies.

9. (Optional) Provide checker parameters. For example, a **save_log_in_part** parameter name, an **=** operation, and a **False** value comprise a **save_log_in_part=False** parameter entry.

   Use the **Click to add new value** and **Remove selected objects from the clipboard** buttons to add or remove checker parameter name, operation, and value data.

10. Click **Finish**.

NX checkers created in the Validation manager honor the **closure rule defined against the NX Check-mate agent**. You therefore need to select the objects defined by the closure rule to run your validation and see the results.

Configure the closure rules for an NX Check-Mate agent

To enable object validation functionality, you must first define the scope of the data to be validated by creating *closure rules* to control the scope of the data to be translated.

The closure rule used by NX validation agents needs to find the NX part item revision objects to undergo NX validation. The system performs the NX validation when the user selects the item, item revision, folder, BOM line, reuse design element, and/or shape design element objects that represent NX parts.

Closure rules are created using the PLM XML application.

Configure the tool for the NX Check-Mate agent

You can specify a tool for the agent when you create an NX Check-Mate agent.

The NX Check-Mate agent uses the scripts defined in the tools to spawn the validation process. You first select the target objects to be validated (folder, item, item revision, reuse design element, shape design element), and then choose some validation checkers under a validation agent.

If the **Is Client Utility** property is set to **False**, the validation request is sent to the server and the server runs a script that spawns the process specified in the validation agent. Otherwise, the validation request is sent to the client machine and the client machine runs the script.

The NX Check-Mate tool is available to use with the NX Check-Mate agent. The tool defines the utility to start the validation process as:

- **ug_check_part.bat** script for Windows
- **ug_check_part.pl** Perl script for Linux

These two scripts are located at **%TC_BIN%** or **$TC_BIN** on the server.

If the **Is Client Utility** property is set to **False**, no further configuration is required.

If the **Is Client Utility** property is set to **True**, the tool defines the **application/ugcheckpart** MIME type. You must perform the following configuration on the client machine so the NX Check-Mate validation process can be run from the client side.

For Windows platform, add the following keys in the Windows registry editor version 5.00:

```
[HKEY_CLASSES_ROOT\.ugcheckpart]
@="ugcheckpart"
"Content Type"="application/ugcheckpart"
[HKEY_CLASSES_ROOT\ugcheckpart]
[HKEY_CLASSES_ROOT\ugcheckpart\shell]
[HKEY_CLASSES_ROOT\ugcheckpart\shell\open]
```

```
[HKEY_CLASSES_ROOT\ugcheckpart\shell\open\command]
@="~Your defined diretory~\\ug_check_part.bat"
```

For the Linux platform, modify the **.mailcap** file under the Teamcenter portal path, which is specified by the **TC_SYSTEM_MAILCAP** system environment variable. Add these two lines to the file:

```
application/ugcheckpart; \
~Your defined directory~/ug_check_part.pl
```

If Teamcenter is running in two-tier mode, you must set the **TC_ROOT** and **TC_DATA** locations in the **ug_check_part** script, which can be specified by the **TC_BIN** environment variable.

Start the validation process for the NX Check-Mate agent

Validation result objects are created by the client application that runs the validation. For example, NX Check-Mate runs the **ug_check_part.exe** file in NX DM mode and creates the validation result objects. The **ug_check_part.exe** file is started indirectly through **ug_check_part.bat** and **ug_check_part.pl** scripts defined in the NX Check-Mate tool for the NX Check-Mate agent.

When you run validations, either the **ug_check_part.bat** or **ug_check_part.pl** script is started and the following arguments are passed into the script:

```
<script name> -role=all -u=<user_name> -p=<password>
-g=<group> -option_file <option_filename>
<validation argument 1> <validation argument 2> <validation argument n>
```

Then the **%UGII_ROOT_DIR%\ug_check_part.exe** script is called with the same set of arguments: **%UGII_ROOT_DIR%\ug_check_part.exe %~1 %~2 %~3 %~4 %~5 %~6 %~7 %~8**.

For example:

```
%UGII_ROOT_DIR%\ug_check_part.exe -u=tcdba -p=tcdbapw -g=dba -option_file
C:\DOCUME~1\x_birada\LOCALS~1\Temp\tcdba021120133579.txt -pim=yes
```

The **option_file** argument is a *user-id/timestamp*.**txt** text file in the temporary directory on the server workstation.

The **validation argument 1**, **validation argument 2**, through **validation argument** *n* are the validation arguments (like **-pim=yes**) defined in the NX Check-Mate validation agent.

The *user-id/timestamp*.**txt** file contains the following data:

- **-checker mqc_check_blend_chamfer_referencing, mqc_pdq_narrow_surface, mqc_report_feature_count**

  The list of checkers is a comma-delineated string after the **-checker** option.

- **-save_result_to_tc always**
- **-tc_file C:\DOCUME~1\x_birada\LOCALS~1\Temp\NX_Check-Mate.txt**

You can manipulate the script to set environment variables and run the **ug_check_part** utility with your own options.

The **NX_Check-Mate.txt** file lists the Teamcenter datasets that have been selected by the closure rule for validation using the NX command line interface (CLI) format. A provided closure rule, **NX Parts**, is used as the NX Check-Mate validation agent.

- Primary parts

  The format for specifying primary parts is:

  @*DB***/***item-id***/***item-revision*

  where:

  - *DB* is the database.
  - *item-id* is the Teamcenter item ID.
  - *item-revision* is the Teamcenter item revision.

  For example:

  @DB/my_part/A

- Nonprimary parts

  The format for specifying nonprimary parts is:

  @*DB***/***item-id***/***item-revision***/***type***/***name*

  where:

  - *DB* is the database.
  - *item-id* is the Teamcenter item ID.
  - *item-revision* is the Teamcenter item revision.
  - *type* is the nonprimary part type.
  - *name* is the name of the nonmaster dataset.

For example:

@DB/my_part/A/specification/my_part-dwg1

For the nonprimary part type, you can use the full type name, abbreviated type name, or single letter type. For example:

@DB/my_part/A/s/my_part-dwg1

Setting up the NX RDDV application

You do not need to explicitly create an NX RDDV agent or checker in Teamcenter when using NX RDDV functionality. The NX RDDV agent is automatically created when Teamcenter is installed. The **ug_check_requirement** checker is created by NX when the RDDV results are saved to Teamcenter from NX.

You cannot start the NX RDDV validation process from Teamcenter. Therefore, the NX RDDV agent does not appear in the **Run Validation** dialog box. **Read** access is required for NX Integration to allow NX RDDV to retrieve and save results from and to the Teamcenter database.

Setting up the BOM Grading application

Create a BOM Grading agent

If the BOM Grading agent was already created, reuse the existing one. You cannot create a new one. Otherwise, create a BOM Grading agent using the following steps.

1. Go to Validation Manager application in Teamcenter and select **Validation Home**.
2. Choose **File→New→Validation Agent**.
3. Select **CondValAgent** and click **Next**.
4. (Optional) Modify the name and provide a description to let users know what the agent contains.
5. (Optional) Modify the following properties:

   - **Keep Overridden Result**

     If **True**, the permanently overridden result is not re-evaluated in the rerun scenario.

   - **Override Reason Is Mandatory**

     If **True**, the designer fills in the override reason as mandatory.

6. Click **Finish**.

Create a BOM grading checker

1. Expand **Validation Home** and select **BOM Grading Agent Revision**.
2. Choose **File→New→Validation Checker**.
3. Select **CondValData** and click **Next**.
4. Type a checker name in the **Name** box.
5. (Optional) Provide a description to let users know what the checker verifies.
6. Click **Next**.
7. Click the **+** button.
8. Select the deployed conditions and assign them to the checker.
9. (Optional) Click the **–** button to remove the unnecessary conditions.
10. Click **Finish**.

Setting up a custom application

To create a custom validation agent to integrate with a custom validation application, you must run the Business Modeler IDE application to create new business objects that are used to instantiate the custom validation agent and validation data objects.

A new subclass of the **ValidationToolAgent** business object must be created for a custom validation agent business object that does not need a tool object. A new tool object provides the information on how a custom validation application is started.

- **ValidationAgent** is a subclass of **Item**.
- **ValToolAgent** is a subclass of **ValidationAgent**.

A new subclass of the **ValData** business object must be created for a validation checker business object. A **ValData** object provides the information on a specific validation functionality that a custom validation application performs.

- **ValData** is a subclass of **Item**.

A new subclass of the **ValidationResult** business object must be created for a validation result business object that provides the information on the result of a validation run performed for a specific custom **ValData** object on a validation target object.

There are three business operations that are defined for a **ValidationAgentRevision** business object. Each custom business object, which is a **ValidationAgentRevision** or **ValToolAgentRevision** business object must implement its own **runValidation** business operation to provide its own validation business logic to create **ValidationResult** objects from the given input objects.

Each custom business object can also specialize the **getValidationTargets** and **getValidationResults** business operations by overriding these methods in a custom class to specialize the business logic, as needed.

```
/**
* This operation performs the validation tasks. Each validation agent revision
must implement this method to process the validation tasks.
* @param valDataRevisionObjects - A vector of ValDataRevision objects that will be
used to validate against the validation targets
* @param selectedObjects - A vector of selected objects that this validation agent
will use to find the actual validation targets for this validation run.
* @param targetObjects - A vector of validation targets for this validation run.
* @return - The status code returned from the run. ITK_ok indicates that the
validation run executed successfully.
*/
int  runValidation(
const std: :vector <tag_t> &valDataRevisionObjects,
const std: :vector <tag_t> &selectedObjects,
const std: :vector <tag_t> &targetObjects,
/**
* This operation returns a vector of validation target object tags via an output
parameter.
* @param valDataRevisions - A vector of ValDataRevision objects that will be used to
limit
the scope of the targets.
* @param selectedObjects - A vector of selected objects that the system will use to find
the actual target objects.
* @param targetObjects - A vector of target object tags that are found from the
given input criteria.
* @return - The return status code. ITK_ok means that the operation successfully
executed.
*/
int  getValidationTargets(
const std: :vector <tag_t> &valDataRevisions,
const std: :vector <tag_t> &selectedObjects,
std: :vector <tag_t> &targetObjects,
/**
* This operation returns a vector of ValidationResult object tags via an ouput parameter.
* @param valDataRevisions - A vector of ValDataRevision objects that will be used to
limit
the scope of the ValidationResult objects.
* @param selectedObjects - A vector of selected objects that the system will use to find
the actual target objects. The resulting target objects will be
*                          used to find the associated ValidationResult objects.
* @param targetObjects - A vector objects that will be used to find the associated
ValidationResult objects.
* @param validationResultObjects - A vector of ValidationResult tags that are found from
```

```
the given input criteria.
* @return - The status code. ITK_ok indicates that this operation successfully executed.
*/
int  getValidationResults(
const std: :vector <tag_t> &valDataRevisions,
const std: :vector <tag_t> &selectedObjects,
const std: :vector <tag_t> &targetObjects,
const std: :vector <tag_t> &validationResultObjects );
```

Validation Manager preferences

Validation data is used as a decision mechanism, ensuring an item revision meets compliance before it is released. To enable object validation functionality, you can set the following site preferences.

- **Baseline_precondition_validation_rule_item_revision**

  Specifies the validation rule item revision used for the **Check_Validation_Results** extension defined for the baseline precondition message.

  This preference is used for baseline operation configuration.

- **TC_ALLOW_ADHOC_VALIDATIONS**

  Determines whether users can set pending validation results to **Pass** or **Fail** on the **Validation Master** form. Some validation utilities may not return either a pass or fail result, so the user may have to make an ad hoc decision regarding the success or failure of the validation. Setting this preference to **TRUE** allows users to set pending results to **Pass** or **Fail**; this is the default setting. Setting this preference to **FALSE** does not allow users to set pending results to pass or fail.

- **TC_VALIDATION_duplicate_nx_valdata_error_on_import**

  Specifies whether an error should be returned when importing an NX validation checker that has the same checker name as an existing NX validation checker in the remote site.

  This preference is used for the Teamcenter multisite configuration.

- **TC_VALIDATION_mandatory_checkers_required**

  Specifies whether the mandatory checkers are required to run when running validation.

- **TC_VALIDATION_requirement_expressions_separator**

  Defines the string used to separate two requirement expressions for display.

- **TC_VALIDATION_requirement_formula_expressions_separator**

  Defines the string used to separate the requirement formula and the requirement expressions for display.

- **TC_VALIDATION_run_validation_hidden_agents**

  Specifies the list of validation agent class names hidden in the list in the **Run Validation** dialog box.

- **TC_VALIDATION_RULE_SET_ITEM_REVISIONS**

  Specifies the list of validation rule item revisions shown in the **Rule** list in the **Rule Evaluation** tab in the **Validation Result Viewer** view.

- **TC_VALIDATION_FILE_TYPES**

  Determines the types of files that can be associated as datasets to validation data or validation results objects. Define the preference by entering the dataset name, **RefName**, and referencers (which are file extensions in the following example) for the types of files that can contain rules or validation results. For example:

  ```
  TC_VALIDATION_FILE_TYPES=
  Text Text *.mqclog
  MSExcel excel *.xls
  ```

  This preference is applicable for NX Check-Mate and NX RDDV validation agents only.

- **TC_VALIDATION_TIMESTAMP_NREFS**

  Compares the timestamp on the validation result and **ImanFile** dataset object to determine whether the result is up-to-date.

3. Managing validation results

Run validations

You can view the generated validation results associated with the target, view the detailed report file stored in the results or delete the selected results. Validation results can also be used as the basis to determine whether a target conforms to standards and can be released.

The **Tools→Validation→Run Validations** menu command lets you specify validation checkers under an agent to verify the selected targets by performing the validation process defined by the agent. The validation process can store the results of the validation process as validation results associated with the selected targets.

1. Select one or more folders, items, item revisions, reuse design element, or shape design elements in the **My Teamcenter** or **Structure Manager** tree.

2. Choose **Tools→Validation→Run Validations** to display the **Run Validations** dialog box.

3. Choose a validation agent from the **Available Validation Agents** shortcut menu.

   If the agent class names are defined with the **TC_VALIDATION_run_validation_hidden_agents** preference, the corresponding agent revisions are hidden from the list.

4. Select one or more validation checkers in the left pane of the **Run Validations** dialog box.

   - If a validation checker item revision is mandatory, it is marked with a red star at the end. If the **TC_VALIDATION_mandatory_checkers_required** preference is set to **True**, after you select an agent revision, all the mandatory validation checkers under it are automatically added into the **Selected Validation Checkers** tree, and they cannot be removed from it unless you select another agent revision.

   - If you select the top agent revision node from the **Available Validation Checkers** tree and click the + button, all validation checkers are added into the **Selected Validation Checkers** tree.

   - If you deselect the top agent revision node from the **Selected Validation Checkers** tree, all the validation checkers nodes under the agent revision are removed from the **Selected Validation Checkers** tree. However, all mandatory checkers are not removed when the **TC_VALIDATION_mandatory_checkers_required** preference is set to **True**.

5. Review the **Selected Objects** pane at the bottom of the window to confirm the objects on which the validation is run.

6. Click **OK**.

The system displays a status message to indicate the validation process has started. If Validation Manager applications are not configured, you are instructed to contact your Teamcenter administrator.

If you have write access to the selected item revision, the system attaches a *validation result summary* to the item revision. This summary lists the result of each specific validation, either **Pass** or **Fail**.

Running validation under single sign-on mode

If you want to run validation under single sign-on mode, you must first install Security Services and configure it on the rich client and the server.

- Configure the *TC_BIN***ug_check_part.bat/pl** file according to the comments inside the file before running validation from the client side or the server side.
- If you want to run validation from the client side, no more configuration is required. Use the standard **ug_check_part.bat/pl** file under the *TC_BIN* directory.

  If you want to run validation from the server side, there are two ways to configure it before running validation:

  - Configure the *TC_BIN***ug_check_part.bat/pl** file according to the comments inside the file.
  - Set the following environment variables before starting the **TCServer** instance. For example, you can add them in *TC_DATA***tc_profilevars.bat** file.

    ```
    set TC_SSO_LOGIN_URL=SSO-login-url
    set TC_SSO_APP_ID=SSO-application-ID
    ```

Viewing validation results

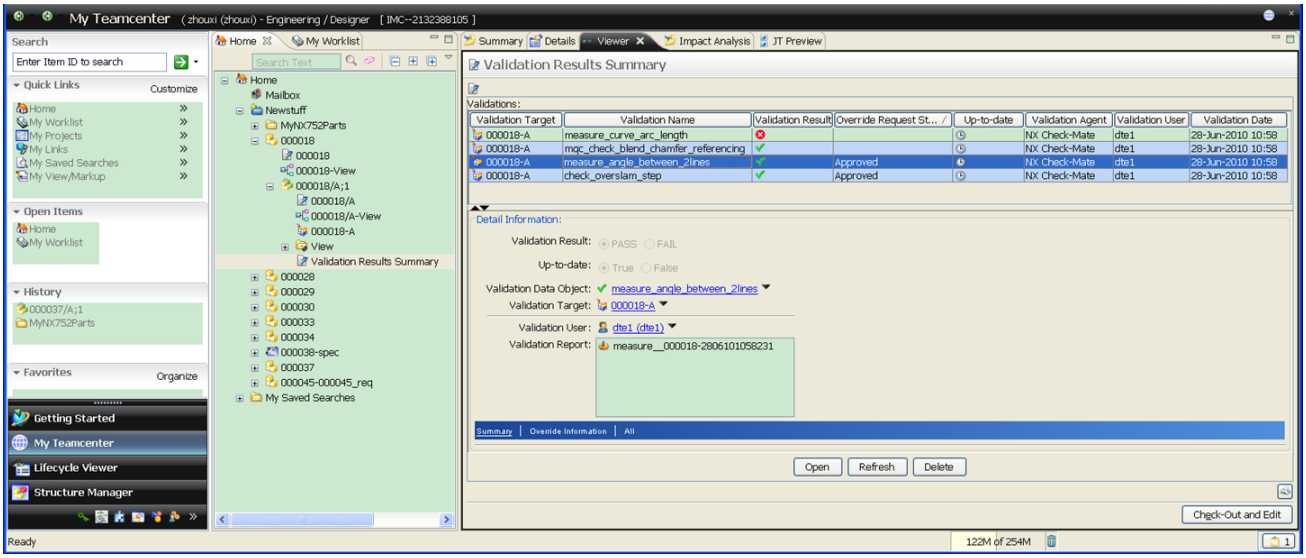
Methods of viewing validation results

Users can view the generated validation results associated with the target, view the detailed report file stored in the results or delete the selected results. Validation results can also be used as the basis to determine whether a target conforms to standards and can be released.

There are three major ways to view validation results:

- **Validation Results Summary**
- **Validation Results Viewer**
- **Report Builder validation reports**

View Validation Results Summary



1. In My Teamcenter, expand the item revision, and open the **Validation Results Summary** object to view all the results for the item revision.

2. Select a validation result from the table.

3. Click **Open**.

   The log content appears automatically unless there are multiple files under the validation result log dataset. If so, the system displays the **Validation Reports** dialog box.

   Some validations may not return report files. Also, a report file must correspond by its file extension to one of the accepted file types defined by the **TC_VALIDATION_FILE_TYPES** preference.

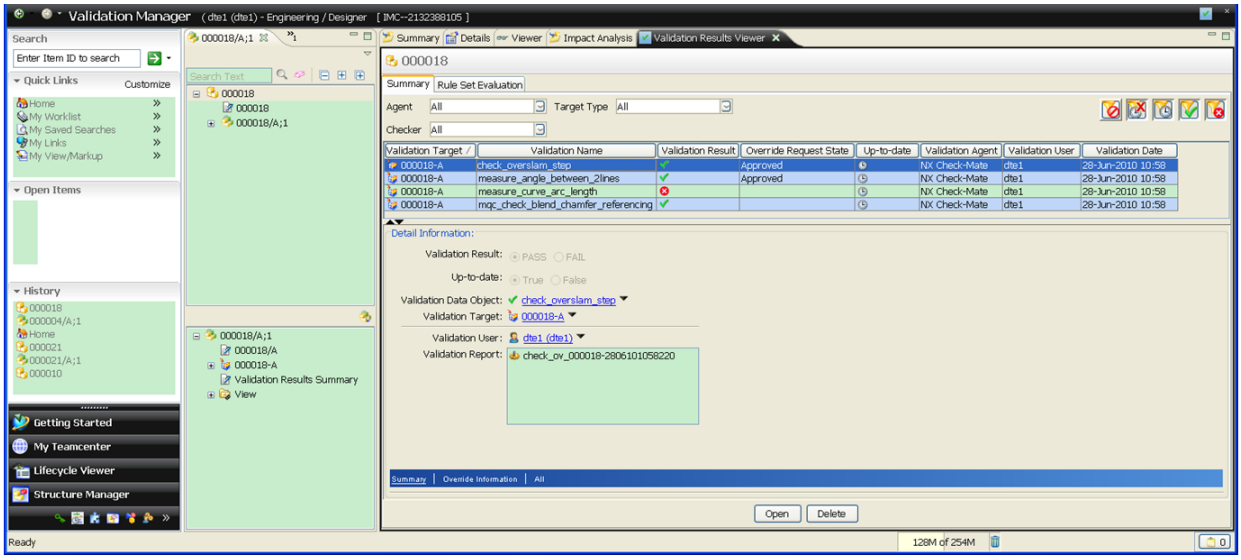
4. If there are multiple files under the validation result, select the report file from the list and click **Open**.

   Validation requirement results are similar to the validation results created by the NX Check-Mate agent, but they are created by the NX RDDV agent, and they contain extra information about the requirements used for checking.

View Validation Results Viewer

Open the Validation Results Viewer view

The **Validation Manager** perspective is shown with two active views:



- The **Selected Workspace Object** view on the left side.
- The **Validation Results Viewer** view on the right side.
  - This view is like the **Summary** view.
  - It contains two tabs to show validation results with or without validation rule file-based evaluation.
  - The viewer pane contains result filters, a result table, and a collapsible information area.
  - You can add or remove columns from the result table at run time.
  - When opened, the detail information area displays detailed information regarding attributes from the selected results in the results table.

When you select a target object from My Teamcenter, you can use four methods to open the **Validation Results Viewer** view:

- In any application, right-click an object and choose **Send To→Validation Manager**.

  The **Validation Results Viewer** view is displayed for the selected object in Validation Manager.

- In My Teamcenter, select an object and choose **Tools→Validation→View Results**.

  The **Validation Results Viewer** view is displayed for the selected object in Validation Manager.

- In Structure Manager, select a BOM line and choose **Tools→Validation→View Results**.

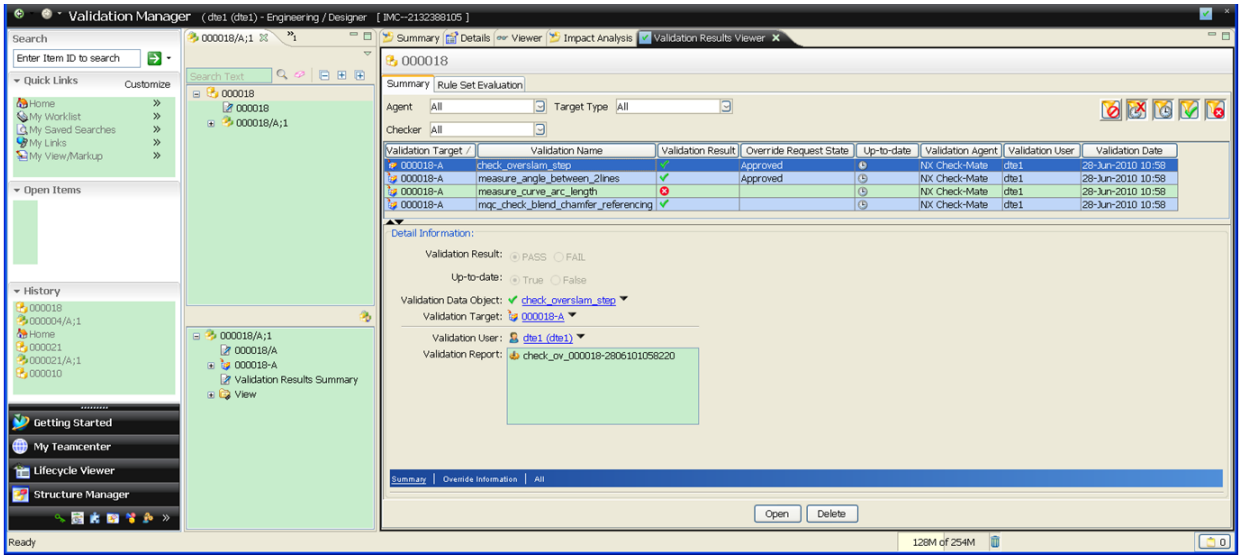
  The **Validation Results Viewer** view is displayed in Structure Manager.

- Click the **HOME** toolbar button in Validation Manager and select an object.

  The associated validation results are displayed in the **Validation Results Viewer** view.

Results from traversing child BOM lines are not provided when you view validation results from the **Structure Manager** window. Only selected BOM lines are listed in the navigator tree in the **Validation Results Viewer** view. To view validation results from the entire Structure Manager tree or subtree, expand the tree in Structure Manager and select all expanded BOM lines.

View results in the Summary tab



Selected Workspace Object tree	Controls the display in the **Validation Results Viewer** view. The tree is a Teamcenter component view similar to My Teamcenter.
List boxes	Provide filter function according to agent name, checker name and target type of results.
Buttons	Enable filter function according to not-run, up-to-date, out-of-date, pass, and fail status of results.
Results data table	Provides specific information about the validation results.
Detail Information pane	Displays a system-defined summary view for a selected validation result (a row in the results table).

By default, the **Validation Results Viewer** view and **Selected Workspace Object** view are not shown in **Validation Manager** perspective. Only when you use these methods to start the **Validation Results Viewer** view to see results can the two views be displayed.

When you select a workspace object or multiple objects from My Teamcenter, and choose **Send To→Validation Manager**, it not only opens the **Selected Workspace Object** view, but also opens the

Validation Results Viewer view. If you select multiple objects and send them to the **Validation Result Viewer** view, multiple **Selected Workspace Object** views are opened just like you send multiple objects to My Teamcenter. The selected folders, items, and revisions are displayed in the **Selected Workspace Object** view similar to My Teamcenter.

Summary view options include:

Option	Description
Agent	Displays the validation agent. Results generated from the selected validation agent are listed in the results table.
Checker	Displays validation checker names. Results generated from selected validation checker (for example, an NX Check-Mate checker) are listed in the results table.
Target Type	Lists the validation target dataset types. Results with selected validation target dataset type are listed in the results table.

Customize the **Summary** view by modifying the validation result summary rendering style sheet. For example, a link to a requirement object can be added and displayed for a selected RDDV result. Click the link and the requirement objects appear in a window.

Summary view buttons include:

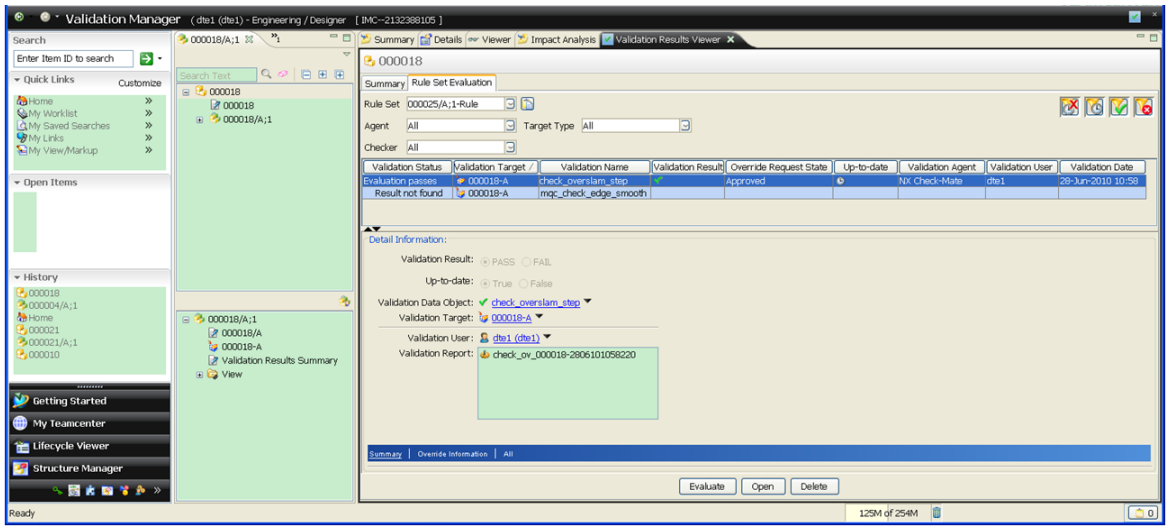
Button	Description
Show Out-of-Date	Sets the display of out-of-date results. Display is enabled by default.
Show Up-to-Date	Sets the display of up-to-date results. Display is enabled by default.
Show Not-Run	Lists all targets that are not validated against selected validation checkers.
Show Passed	Sets the display of items that pass validation.
Show Failed	Sets the display of items that fail validation.

You can add or remove columns, sort tables per column, and so on in the results data table. The results data table includes:

Data	Description
Validation Target	Lists the datasets associated with the item revision for which the validations were performed.
Validation Name	Displays the checkers that are run.
Validation Agent	Lists the validation agent. A check that belongs to this validation agent is performed against the target object, for example, the NX Check-Mate agent.

Data	Description
Override Request State	Indicates the **override status** of the validation result: **Pending**, **Approved**, or **Rejected**.
Validation Date	Lists the date and time each validation was performed.
Up-to-date	Specifies whether validation was run on the current target object.
Validation Result	Indicates a **Pass** or **Fail** result for each validation performed. If the current result is **overridden** from fail to pass, it shows pass, and vice versa.
Validation User	Displays the validation user ID string.

View results in the Rule Set Evaluation tab



Rule Set

Enter a validation rule item revision from My Teamcenter or from the Teamcenter search results.

or

Type a validation rule item revision (standard format: *item-id/revision-id*; if you use multifield keys, the format may look like *item-id,object-type,revision-id*, depending on how the key is defined for that business object type).

The selected validation rule file is called to evaluate validation results.

Include the validation rule item revisions in the **TC_VALIDATION_RULE_SET_ITEM_REVISIONS** preference to list them in the **Rule Set** menu.

Validation Status

Displays results status from the validation rule file evaluation. This status may be different from the status that originally

	returned from the Validation Manager application. A result with up-to-date status or failed status from the Validation Manager application may pass a validation rule file evaluation if the check is defined as nonmandatory in the validation rule file.
Evaluate button	Unavailable by default, this button is available when the **Workspace** objects from the left navigator tree and validation rule file are selected. Click this button to evaluate validation results.

The validation rule file is defined using validation rule XML format and saved as the **ValidationRuleSet** dataset. Attach the **ValidationRuleSet** dataset to an item revision with the **TC_Validation** relation. There is to be one **ValidationRuleSet** dataset under one item revision.

When no validation rule file is available, or you do not want to select any validation rule file, view validation results from the **Summary** view.

The evaluated status descriptions include:

Status	Description
Evaluation passes	The validation results pass the rule evaluation.
Result out of date due to part changed	The part **Last-Modified Date** is later than the date that the validation was performed, and the result was saved into Teamcenter.
Result failed	The validation result status fails for a mandatory check.
Parameter verification failed	The validation passes and is up-to-date, but the parameter verification fails. Parameters listed in the validation rule are verified by first comparing the name and value pairs with the **validation_parameters** attribute of **validation result** object and then the log file if the first step returned a fail decision.
Result not found	The validation result is not found. The validation is not performed or the result is not saved into Teamcenter.
Missing result log file	The validation result log dataset or file is missing. This failure is checked only when a validation rule contains a parameter condition.

Report Builder validation reports

There are three validation summary reports and one item report provided by Teamcenter. Summary reports present information based on a saved query definition. Item reports are run on one or more selected items based on a class of items. The validation summary reports are:

- **VO - NX Validation Result on Item Revisions**

Displays all NX validation results on the item revisions specified in the search criteria.

- **VO - Validation Result**

  Displays all validation results from the specified search criteria.

- **VO - Validation Result with Override Requests Information**

  Displays all validation results from the specified search criteria, including the override requests information. It is similar to **VO - Validation Result** report but includes more information about override information.

The item report is:

- **VO - Item Revisions Validation Results**

  Displays all NX Check-Mate validation results on the selected item revisions.

Override validation failure

Who can override validation failure?

For a failed validation, any user with sufficient privileges to write to the validation result object can request to override the validation failure result. Designated users can override a failed result by using shortcut menu commands in the validation results viewer view or by using a workflow. To approve a result override request, a user must belong to the validation administration group and the override approver role.

The Teamcenter administrator must use the Organization application to add members to the validation administration group and the override approver role.

Users with sufficient privileges can:

- Create an override request.
- View an override request.
- Edit an override request.
- Delete an override request.

Create an override request

Any user with write access to a validation result object can create an override request.

1. Right-click the validation result from the **Validation Results Viewer** view and choose **Override→Create**.

   The system displays the **Create Override** dialog box.

2. Select appropriate options in the **Create Override** dialog box.

   a. Select an option from the **Type** list.

Permanent	The approved override request always applies to the validation result until the original request user deletes the override request.
Temporary	The approved override request applies only to the validation result until the test reruns and validation result updates. The system automatically removes the approved override request and updates the validation result after the test reruns.

   b. Select an option from the **To State** list.

   c. Select an option from the **Category** list.

   > Note:
   >
   > **Category**, **Brief Reason**, and **Detailed Reason** values are optional unless the administrator selected **Override Reason Is Mandatory** in the **New Agent** dialog box when creating an agent object.

   d. Select an option from the **Brief Reason** list.

   e. Enter a reason for the override request in the **Detailed Reason** box.

   f. (Optional) Select a proposed decision user from the **Override Decision User** list in the **Create Override** dialog box and **Edit Override** dialog box. The **Override Decision User** list shows all the users under the **Override Approver** role in the **Validation Administration** group. This box is optional so you can leave it blank. If you select a user from the **Override Decision User** list or change to a different decision user, notification mails are sent to the current selected decision user and/or the previous selected decision user.

   If your company's best practice is to approve or reject an override request from the Validation Manager application, specify a user from the **Override Decision User** list when creating a request, so the suggested approver is notified to review the request. If your company's best practice is to approve or reject an override request in a workflow process, leave the **Override Decision User** box blank, because the workflow already has a separate notification mechanism.

3. Click **OK**.

   - The system displays **Override Request Status** as **Pending**.

- The **Result** column shows the original validation result status.

View an override request status

1. Select one or more validation target objects.

   A validation target can be any workspace object, such as an item, item revision, dataset, folder, reuse design element, and shape design element, among others.

2. From the shortcut menu, choose **Send To→Validation Manager**.

3. Check the **Override Request State** column for the override status of the validation result.

   - **Pending**

     The override is awaiting review.

   - **Approved**

     The reviewers accept the override. The **Result** column shows the override result.

   - **Rejected**

     One or more of the reviewers deny the request. The **Result** column shows the original validation result.

4. View the detail status and provide comments in the **Detail Information** box.

   a. Select one result in the **Results Viewer** view.

   b. View the information in the **Detail Information** box.

Edit an override request

Only the user who made the original request can edit an override request.

1. Select one validation result with a **Pending**, **Approved**, or **Rejected** override status in the **Validation Results Viewer** view.

   For override requests with **Approved** or **Rejected** status, when you edit the override request, the system resets the request to **Pending** status.

2. From the shortcut menu, choose **Override→Edit**. The system displays the **Edit Override** dialog box, which contains the same information as the **Create Override** dialog box.

Delete an override request

Only the original request user can delete an override request.

1. Select one validation result with a **Pending**, **Approved**, or **Rejected** override status in the **Validation Results Viewer** view.

2. From the shortcut menu, select **Override→Delete**. The **Delete Override** dialog box is displayed.

3. Click **Yes**.

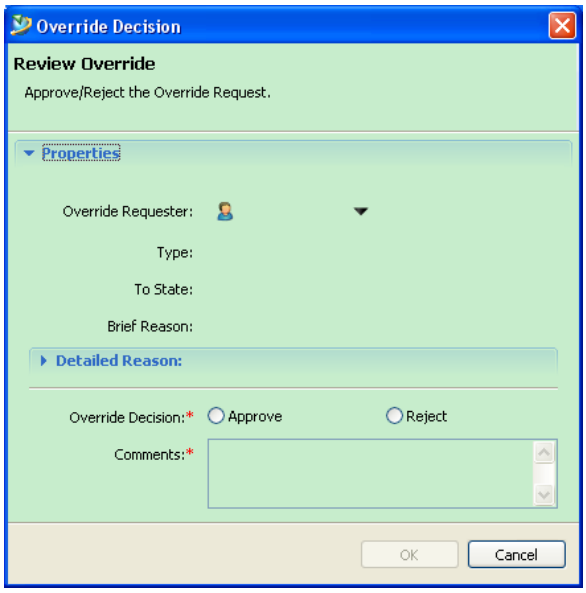
Review an override request using interface commands

To approve or reject an override request, the reviewer must belong to the validation administration group and the override approver role.

1. Select one validation result with a **Pending**, **Approved**, or **Rejected** override status in the **Validation Results Viewer** view.

2. From the shortcut menu, choose **Override→Review**.

   The system displays the **Override Decision** dialog box.



3. Select **Approve** or **Reject**.

4. Enter comments for the override decision in the **Comments** box.

5. Click **OK**.

The updated **Validation Results Viewer** view appears.

- When the override request is approved, the **Override Request State** column indicates the **Approved** status, and the **Result** column displays the overridden result.
- When the override request is rejected, the **Override Request State** column indicates the **Rejected** status, and the **Result** column displays the original status.

Review an override request in a workflow

Create a result override approval process

1. Choose **New→Process**.
2. Select **Result Override Approval** from the **Process Template** list.
3. In **My Worklist**, perform the **select-signoff-team** task.

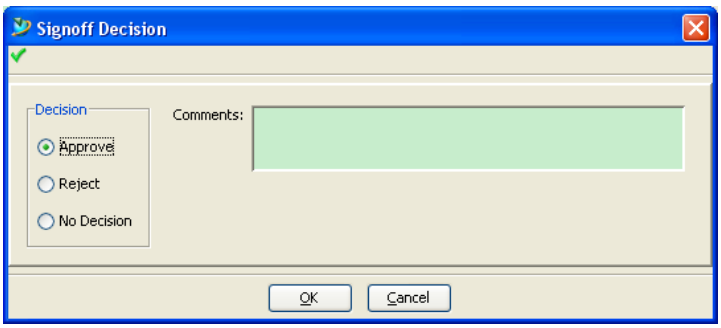
   Signoff team members must belong to the validation administration group and the override approver role.

After the select-signoff-team task is complete, the process continues in the workflow and members of the signoff team receive the signoff task.

Perform an override approval task in a result override approval process

1. In **My Worklist**, select the **perform-signoff** task.
2. Choose **Actions→Perform**.

   The system displays the **Signoff Decision** dialog box.



3. Select **Approve** or **Reject** or **No Decision**.
4. Enter comments regarding the signoff decision in the **Comments** box.

5. Click **OK**.

Result override notification

The **Override Approver** can approve or reject an override request from either the Validation Manager application, or a workflow process. When the override requester specified an **Override Decision User** when creating or updating an override request, notification mails are sent to corresponding participants when an override is requested, updated, approved, rejected, or deleted.

> Note:
>
> By default, validation override notifications are sent through operating system mail. To send notifications through Teamcenter mail, set the **TC_VALIDATION_send_envelope_for_override** preference to **TRUE**.

- An email notifying the approver of an override request includes these messages to either approve or reject the request:

```
Send the attachment to the Validation Results Viewer
Right click on this result.
Choose Override→Review.
```

- An email notifying the old approver and the new approver that their assignments have changed includes this message:

```
You are no longer the approver of this override request.
```

- An email notifying the requester that the override request has been approved includes this message:

```
Your override request is approved by [Approver].
```

- An email notifying the requester that the override request has been rejected includes this message:

```
Your override request is rejected by [Approver].
```

- An email notifying the approver and the requester that the override request has been deleted includes this message:

```
The override request is deleted.
See comment on the Result Detail panel.
```

- An email notifying the approver that the override request has been updated includes this message:

```
The override request is updated by [Requester].
```

Validation results in a workflow or baseline operation

To ensure qualities in design process, three workflow handlers are defined and can be used to evaluate validation results of selected targets based on overall status or a validation rule before releasing the targets in a workflow.

- **VAL-check-validation-result**
- **VAL-check-validation-result-with-rules**
- **VAL-set-condition-by-check-validation-result**

The **Check_Validation_Results** extension is another extension that is registered into a precondition message for the baseline operation. It can be used to verify that all parts pass the validation rule conditions before the baseline is created.

These three handlers and the extension are applicable to the validation results generated by the NX Check-Mate and NX RDDV validation agents.

The **ResultOverrideApproval** workflow template can be used to approve or reject the override requests on the validation results associated with the workflow targets.

This template can be used independently or within any customer's workflow templates for privileged users to approve all override requests associated with the workflow targets.

In the **ResultOverrideApproval** workflow template, the **overridecondition** task has a **Start** action handler (**VAL-set-condition-result-overrides**) that sets the condition to **EPM_RESULT_True** if there are pending result override requests for the workflow targets. If there are any pending result override requests, the workflow continues to **OverrideReviewTask** for privileged users to signoff; otherwise, the workflow continues to **OverrideOrTask**, then completes.

The **OverrideReviewTask** has **Validation Administration/Override Approver/1** as the default signoff profile. Two action handlers (**VAL-approve-result-overrides** and **VAL-reject-result-overrides**) are used to set the approval state of the associated overrides whose current state is not approved. When triggered, the **VAL-approve-result-overrides** action handler sets all requested overrides to an approved state, while the **VAL-reject-result-overrides** action handler sets all requested overrides to a rejected state.

Copying validation results

Methods of copying validation results

There may be times you want to ensure both the target dataset and its associated validation results are copied as object when an item or item revision is revised, baselined or copied.

However, by default, the validation results summary and associated validation results are not copied when an item revision is revised, baselined, or copied.

Also, by default, if the target dataset of a revised, baselined, or copied item revision is copied as object or as reference:

- Its validation results summary and associated validation results are not copied.
- The item revision after revise does not display the validation results summary.

There are two methods you can use if you want to copy as object the validation results summary and the associated validation results when the target dataset is copied as object. You must use one method or the other; you cannot use both.

- Siemens Digital Industries Software recommends that you set the deep copy rule for the **ItemRevision** business object for the **ValidationMaster Form related object to CopyAsObject** for the **TC_Validation** relation type.
- An alternative method, which is not recommended, is to set the deep copy rule for the **ValidationMaster Form** related object to **NoCopy** and use the **predefined Copy_Validation_Results extension**.

Copy validation results by setting the deep copy rule for ValidationMaster Form to CopyAsObject

Siemens Digital Industries Software recommends this method for copying validation results.

1. In the Business Modeler IDE, open the **ItemRevision** business object, click the **Deep Copy Rules** tab, and verify the following settings for the **UGMASTER** attached business object:

   - The **Operation Type** is **Revise** or **Save As**.

     This ensures the target dataset is copied as object when an item revision is revised, baselined, or copied.

   - The **Condition** is **isTrue**.
   - The **Action** is **CopyAsObject**.

2. On the **ItemRevision** tab, add or edit the deep copy rules for the **ValidationMaster Form** business object with the following settings:

   - The **Operation Type** is **Revise** or **Save As**.

     This ensures the validation result summary is not copied as reference but re-created based on the copied validation results.

- The **Condition** is **isTrue**.
- The **Action** is **CopyAsObject** for the **TC_Validation** relation.

The **Copy_Validation_Results** extension must not be added to **ITEM_deep_copy** operation.

Copy validation results by assigning a Business Modeler IDE predefined extension

> Caution:
>
> The following is the legacy method and is not recommended. Support for this legacy method may be removed from a future Teamcenter version.
>
> Instead, Siemens Digital Industries Software recommends that you **set the deep copy rule** for the **ValidationMaster Form** business object to **CopyAsObject**.
>
> You must use either that method or the following one; you cannot use both.

Only the Teamcenter administrator can enable the **Copy_Validation_Results** extension used in this procedure.

1. In the BMIDE for Data Model Design, open the **ItemRevision** business object, click the **Deep Copy Rules** tab, and verify the following settings for the **UGMASTER** attached business object.

   - The **Operation Type** is **Revise** or **Save As**.

     This ensures the target dataset is copied as object when an item revision is revised, baselined or copied.

   - The **Condition** is **isTrue**.
   - The **Action** is **CopyAsObject**.

2. On the **Deep Copy Rules** tab, verify the following settings for the **ValidationMaster Form** attached business object.

   - The **Operation Type** is **Revise** or **Save As**.

     This ensures the validation result summary is not copied as reference but re-created based on the copied validation results.

   - The **Condition** is **isTrue**.
   - The **Action** is **NoCopy** for the **TC_Validation** relation.

3. On the **Item Revision** object **Operations** tab, add the **Copy_Validation_Results** extension to the **ITEM_deep_copy** operation as a postaction. The condition is **isTrue**.

   The extension copies the results that are up-to-date and ignores the outdated results at the time when the extension is executed.

Import and export validation results using PLM XML

Transfer mode objects are created in the PLM XML application and are displayed as options when users import or export objects or system data using one of the Teamcenter applications. They allow users to import and export data.

Transfer mode objects are made up of rules that configure the import and export operations. To enable validation result PLM XML import export functionality, a top-level import/export closure rule and transfer mode must be defined to handle the validation results import and export.

In previous versions, the validation results were exported and imported associated with a validation target dataset. A transfer mode using the **VALIDATION_PIE_CONTEXT_STRING** context was required to export validation results along with the validation target dataset.

The validation result is not restricted for export and import associated with the validation target dataset, and the **VALIDATION_PIE_CONTEXT_STRING** context was deprecated. Instead, any transfer mode using the **DEFAULT_PIE_CONTEXT_STRING** context can export and import the validation result if the closure rules can find the validation result objects.

The **ExportValidationResult** transfer mode can be used to export validation results associated with the item revision. The **incremental_import** transfer mode can be used to import validation result included in a PLM XML file. Both use the **DEFAULT_PIE_CONTEXT_STRING** context.

> Note:
>
> The **Validation Result** class can be extended from the Business Modeler IDE. If the customized validation result objects have specific business logic, Siemens Digital Industries Software recommends that you create corresponding transfer modes to export and import them.

Troubleshooting Validation Manager issues

Validation report file display issues

If you cannot display the validation report files:

- The defined values of the **TC_VALIDATION_FILE_TYPES** preference do not include the type of file produced by this validation. In this case, there are no report files stored with the validation results.
- The report files may not appear if environment variables are not defined properly to display named references.

Change item revision ownership makes Validation Results Summary invisible

Changing the ownership of an item revision makes the **Validation Results Summary** invisible for the new owner. To correct the problem, perform the following steps.

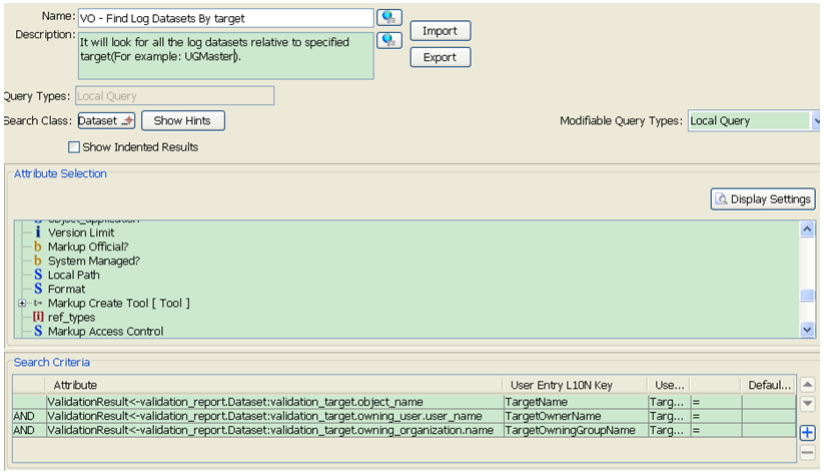
To avoid this issue, set the **TC_VALIDATION_unify_itemrev_validation_form_ownership** preference to **TRUE**.

1. Log on as the new owner, in the rich client choose **Edit→Options**, and in the **Options** dialog box, choose **General→Item Revision**. Move **Validation** from the **Available Relations** list on the left to the **Shown Relations** list on the right.

2. If the previous step does not resolve the issue, check access rules in the Access Manager at your site. Verify whether there is an access rule that specifies that the new owner cannot view objects owned by others or not owned by the new owner's group. If the access rule does not exist, the original owner must select all the components under the item revision including **Validation Results Summary** when changing ownership. This makes the **Validation Results Summary** visible to the new owner.

3. If the new owner cannot open the individual validation result because the result log dataset cannot be read by the new owner according to the access rule, perform the following.

   Use the Query Builder application to create a saved query to find all the log datasets in the validation result related to the target dataset. Log on as the original owner, run the saved query, and find all log datasets related to the target dataset under the item revision, and then change ownership of the log datasets to the new owner.



Cannot save NX validation results to the Teamcenter database nor retrieve results from it

Read access to Teamcenter NX checker item and revision objects for NX validation users is required for NX Integration to allow NX validation applications, such as NX Check-Mate and Requirement Validation (NX RDDV), to retrieve and save results from and to the Teamcenter database.

If NX validation failed to work correctly due to denied read access to checker item or checker revision objects, do one of the two following alternatives:

- Modify the Access Manager rules to grant read access to users for the following classes:
  - **NXRDDVValDataRevision**
  - **NXCMValDataRevision**
  - **NXRDDVValData**
  - **NXCMValData**
- Change ownership of the checker item or checker revision objects to a user with Teamcenter administration privileges.

  1. The database administrator displays Validation Manager in the rich client and fully expands the NX Check-Mate item. There are a number of checker item and checker revision objects under the NX Check-Mate revision object.

  2. The database administrator changes the checker item and checker revision ownership to the new user with Teamcenter administration privileges by selecting the checker item and checker revision objects to process and clicking the **Take Ownership** toolbar button.

  After the ownership of checker objects are transferred to the new user, the default Access Manager rules allow read access to the objects.

4. Validation rules

Overview of validation rules

Validation rules define the criteria needed to verify that all required checkers were executed against a certain target, and that the expected results were achieved.

For each checker, you can define the following criteria:

- The test to be performed
- The target to be tested
- The results to expect

Validation rules are defined in a validation rule XML file. Each rule is made up of the following information:

- A validation checker name is required. This is the unique name of a validation checker, for example, **mqc_examine_geometry**.
- A **mandatory** flag is required. There are two options for this flag:
  - If set to **True**, the checker must run and pass.
  - If set to **False**, the checker must run, but the result is ignored.

    It is **False** by default.
- An event is optional. An event defines a situation in which a rule should be included, or a situation in which a rule should be excluded, and when the rule is applied.

  To determine the situation:

  - Use Teamcenter properties. The property of a tested item revision can be used as a condition to decide whether the rule is applicable or not.
  - Use an external argument. An external argument value can be verified to decide whether the rule is applicable or not.

  The events are optional for a rule. A rule without events is still valid. One rule can contain multiple events, or it does not define events. If multiple events are defined for a rule, this rule can be applied only when all the events are qualified.

- A target is required. The type of validation target object could be used to decide whether to select the rule or not. It is required.

- The checker parameters are optional. This is a list of parameter and value pairs, validated against the parameters contained in a validation result. The check parameters can be contained in the validation parameter object of a validation result, or come from the validation report dataset file.

Validation rules are widely used in Workflow and Validation.

The Validation Rule Editor (a Java program) can be used interactively to create or modify rules in a validation rule file.

There are examples in the **TC_DATA** directory as well. You can use them to help you understand the validation rule.

Using validation rules in a baseline operation

Validation rules can be leveraged in baseline operations to ensure design qualities. Target parts are verified according to the specified validation rule item revision before the baseline is created.

The **CheckValidation Results** extension is registered into a precondition message from the baseline operation. To verify that all parts pass the validation rule conditions before the baseline is created, the Teamcenter administrator must enable this extension and set the validation rule item revision to the **Baseline_precondition_validation_rule_item_revision** preference.

If all target datasets pass the validation rule verification, the baseline is created. Otherwise, the baseline operation is blocked.

Using validation rules in a workflow

Validation rules can be leveraged in Workflow to ensure design qualities. Target parts are verified according to the specified validation rule item revision before Workflow moves to the next step.

The **VAL-check-validation-result-with-rules** handler can be registered to any workflow task and returns **EPM_go** if all target datasets pass the validation rule set verification. Otherwise, **EPM_nogo** is returned and the workflow is blocked.

The **VAL-set-condition-by-check-validation-result** action handler can be registered to the workflow condition task to set conditions based on the result of verification of the validation rule. The condition is set to **True** if all target datasets pass the validation rule verification. Otherwise, the condition is set to **False**.

Using validation rules for viewing results

Validation rules can be used in the **Rule Set Evaluation tab** when viewing the results using the **Validation Results Viewer** view. When using the same rule as the one used in Workflow and baseline

operations, you can preview the evaluation results before sending the item revisions to Workflow and baseline operations.

Using validation rules for importing results in NX parts

Validation rules can also be used to import validation results saved in an NX part file while importing an NX assembly from a native file system into Teamcenter.

For more information, see the utility section in *Teamcenter Integration for NX* in the NX documentation set.

Set up the Validation Rule Editor

You can create the validation rule XML file manually using the preferred editing tools, or create it using the Validation Rule Editor included with the NX product.

In Teamcenter, the rule set file is saved as the **ValidationRuleSet** dataset. You can use the Validation Rule Editor to view and edit the rule set file.

The Validation Rule Editor is a Java program included with the NX product. **validation_rule.bat** for Windows and **validation_rule** for Linux are two script files under the **TC_BIN** location.

The opening/editing program of the **ValidationRuleSet** dataset is associated to these script files. When you open the **ValidationRuleSet** dataset from the rich client, a Validation Rule Editor starts a script.

To make the script work properly, the system must know where NX is installed and the MIME or MAILCAP settings. The following configuration must be done before running the Validation Rule Editor.

Windows systems:

1. Set the **UGII_BASE_DIR** environment variable in your system.

   Or, find this line in the **validation_rule.bat** file in the *TC_ROOT***bin** directory:

   ```
   rem set UGII_BASE_DIR=~Your UGII_BASE_DIR~
   ```

   Change it to:

   ```
   set UGII_BASE_DIR= D:\NX7.5
   ```

2. Add the following keys in the windows registry. These keys tell the Windows system the MIME type of the validation rule set, file extension, and the open and edit commands.

   When you open the **ValidationRuleSet** dataset, Teamcenter finds the open/edit command by MIME type.

```
[HKEY_CLASSES_ROOT\.vrx] @="validationruleset"
"Content Type"="text/validationruleset"
[HKEY_CLASSES_ROOT\validationruleset]
[HKEY_CLASSES_ROOT\validationruleset\shell]
[HKEY_CLASSES_ROOT\validationruleset\shell\edit]
[HKEY_CLASSES_ROOT\validationruleset\shell\edit\command] @="\"~Your
TC_BIN~\
\validation_rule.bat\" \"%1\""
[HKEY_CLASSES_ROOT\validationruleset\shell\open]
[HKEY_CLASSES_ROOT\validationruleset\shell\open\command] @="\"~Your
TC_BIN~\
\validation_rule.bat\" \"%1\""
```

Linux systems:

1. Set the **UGII_BASE_DIR** environment variable in your system.

   Or, find this line in the in the **validation_rule.bat** file in the *TC_ROOT*/**bin** directory:

   ```
   rem set UGII_BASE_DIR=your_NX_base_dir ;
   ```

   Change it to:

   ```
   set UGII_BASE_DIR=NX7.5
   ```

2. Add two lines to the **.mailcap** file under Teamcenter portal path which is specified by the **TC_SYSTEM_MAILCAP** system environment variable.

   ```
   text/validationruleset; \ ~Your TC_BIN~/validation_rule
   ```

Validation rule file formats and examples

Introduction to validation rule file formats and examples

The validation rule file is an XML file using PLM XML format. Use the Validation Rule Editor to view or modify the rule file in PLM XML format to ensure the data's integrity in a validation rule file.

Validation rule file in PLM XML format

The **ValidationRuleSet** element is optional and contains multiple rules.

- If not defined, all the rules defined in the file are selected to evaluate the validation results.
- If defined, it should match a specified rule set you want to verify. For example, to verify a couple of geometry rules, you can input a **Geometry_Rules** argument. This argument is used to search a

validation rule set that has the **Geometry_Rules** name. Once the **ValidationRuleSet** element is found in the rule file, all the included rules are verified.

When the **ValidationRule** element defines a rule:

- The validation rule name is optional.
- The validation checker name is required.
- It is case insensitive and supports wildcards.
  - ? represents any one character.
  - * represents zero or more characters.

The **Mandated** element requires a mandatory flag:

- **True** requires the checker to run and pass.
- **False** requires the checker to run but the result is ignored.

The **eventRefs** element is optional and references a **ValidationEvent** element.

The **parameterRefs** element is optional and references a **CheckerParamter** element.

The **target** subelement is required and references a **Target** element.

The **ValidationEvent** element defines an event for rule.

- **name** is optional.
- The **eventClass** clause indicates a **ItemRevision** property.
  - **ITEMREVISION:property_name** is optional.

    The target item revision property value should be tested against the **ValueList** subelement.
- **applyTo** is optional and reserved for an application other than Teamcenter. Currently, it is used in NX to represent the part attribute or part name. The format is **PARTATTRIBUTES**:*attribute_name* or **PARTNAME**.

  The target part attribute or part name should be tested against the **ValueList** subelement.
- **application** is required and indicates this event is used in Teamcenter or other applications.
- **exclude** is required.

- **True** means the rule is not applied if the event condition is matched.
- **False** means the rule is applied if the event condition is matched.

  False is the default value.
- The **ValueList** subelement provides a list of values to be verified.
  - It is case insensitive and supports wildcards.
  - ? represents any one character.
  - * represents zero or more characters.

- Subelements:
  - **ValueList** is a list of values to be verified.

The **Target** element defines the target for a rule.

- **targetType** is optional and provides the class type of the target validation object.
  - The **Item** subelement lists subtypes of the validation target object.
    - It is case insensitive and supports wildcards.
    - ? represents any one character.
    - * represents zero or more characters.

The **CheckerParameter** element defines the parameter and value pair.

- **name** is required.
- **value** is required.
- **operator** is the compare operator between name and value and is required.

Validation rule file in legacy XML format

When you create a validation rule file in the legacy XML format, use the following guidelines:

- Rule set node attributes are required and include **Name**, **Description**, **User**, and **Date**. These attributes are mapped as Teamcenter validation rule file object attributes when the XML rule is imported.

- The rule node defines a rule.
  - The checker name is a validation checker name. The **Checker** node is required.
  - When a check is mandatory, the check must run and must pass. When a check is optional, the check must run and result status is ignored. The **mandated** node is required.
  - The applicable dataset types list must be defined. The **Datasets** node is required.
  - The applicable events list contains the event values to which the check is applicable. When no event value list is defined, the rule is applicable at all times. The event value can contain * as a wildcard. The values listed can be exclusive when exclude is set to **Yes**.
  - When the class attribute is defined for an event list, the class clause defines which target revision attribute value should be tested against the event values list.
  - Parameters are first verified from the **Validation Result** object parameters. If a negative decision is returned, the log file is opened to verify again. For the NX Check-Mate log, the parameter values saved in the profile level are verified (values listed at child checker levels are ignored). For the RDDV log, because there is no child checker, all the parameter values listed are verified.
  - If a negative decision is returned, the log file is opened to verify again. For the NX Check-Mate log, the parameter values saved in the profile level are verified (values listed at child checker levels are ignored). For the RDDV log, because there is no child checker, all the parameter values listed are verified.
- The order in which rules are defined in the XML rule file does not impact results.
- The same checker can be instanced in the definition of multiple rules in the same rule file.

Sample validation rule file for NX Check-Mate results

There is a **validation_rule_sample_file.vrx** file example at **%TC_DATA%** for verifying the results of NX Check-Mate agent. This sample rule file is in the new PLM XML format.

```
<?xml version="1.0" encoding="utf-8"?>
<PLMXML xmlns="http://www.plmxml.org/Schemas/PLMXMLSchema"
        xmlns:rule="http://www.plmxml.org/Schemas/
PLMXMLValidationRuleSchema"
        schemaVersion="6"
        date="2008-06-25"
        time="10:40:36"
        author="wuyun" >
<rule:ValidationRuleSet id="id1" ruleRefs="#id3 #id4 #id5 #id6 #id7"
 name="Test_rule_set_1" description="Validation Rule Set example" />
<rule:ValidationRuleSet id="id2" ruleRefs="#id8" name="Test_rule_set_2"
```

```
 description="Validation Rule Set example 2" />
<rule:ValidationRule id="id3" name="rule 1"
checker="mqc_check_dim_attach_solid"
        mandated="true"
        eventRefs="#id9">
        <rule:Target id="id17" type="string" targetType="Dataset">
           <Item value="UGMASTER" />
      <Item value="UGPART" />
    </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationRule id="id4" name="rule 2"
checker="mqc_examine_geometry"
        mandated="true">
    <rule:Target id="id18" type="string" targetType="Dataset">
      <Item value="UGMASTER" />
      <Item value="UGPART" />
    </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationRule id="id5" name="rule 3"
checker="mqc_check_dim_attach_solid"
        checkerVersion="3"
        mandated="false" >
    <rule:Target id="id19" type="string"  targetType="Dataset">
      <Item value="UGPART" />
    </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationRule id="id6" name="rule 4"
checker="mqc_exam_geom_combo"
        mandated="false"
        eventRefs="#id10"
        parameterRefs="#id14 #id15">
    <rule:Target id="id20" type="string" targetType="Dataset">
      <Item value="UGMASTER" />
      <Item value="UGPART" />
      <Item value="UGALTREP" />
    </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationRule id="id7" name="rule 5"
checker="mqc_examine_geometry"
        mandated="false"
        eventRefs="#id11"
        parameterRefs="#id16">
    <rule:Target id="id21" type="string"  targetType="Dataset">
      <Item value="UGMASTER" />
      <Item value="UGPART" />
    </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationRule id="id8" name="rule 6"
checker="mqc_exam_geom_combo"
```

```
        mandated="false"
        eventRefs="#id12 #id13" >
    <rule:Target id="id22" type="string" targetType="Dataset">
      <Item value="UGMASTER" />
      <Item value="UGPART" />
      <Item value="UGALTREP" />
      </rule:Target>
  </rule:ValidationRule>
  <rule:ValidationEvent id="id9" eventClass=
"ITEMREVISION:owning_group" name="Event 1"
        exclude="true" >
    <rule:ValueList id="id23" type="string">
      <Item value="chassis.pe.company" />
      <Item value="powertrain.pe.company" />
    </rule:ValueList>
  </rule:ValidationEvent>
  <rule:ValidationEvent id="id10" applyTo="PARTATTRIBUTES:CHECK_FLAG"
 exclude="false" name="Event 2" application="NX">
    <rule:ValueList id="id24" type="string">
      <Item value="*" />
    </rule:ValueList>
  </rule:ValidationEvent>
  <rule:ValidationEvent id="id11" name="Event 3">
    <rule:ValueList id="id25" type="string">
      <Item value="S*" />
      <Item value="Status*3" />
      <Item value="Status_4" />
    </rule:ValueList>
  </rule:ValidationEvent>
  <rule:ValidationEvent id="id12" exclude="false" name="Event 4"
        rule:ValueList id="id26" type="string">
      <Item value="Assembly" />
      <Item value="Modelling" />
    </rule:ValueList>
  </rule:ValidationEvent>
  <rule:ValidationEvent id="id13" name="Event 5">
    <rule:ValueList id="id27" type="string">
      <Item value="Milestore 1" />
    </rule:ValueList>
  </rule:ValidationEvent>
  <CheckerParameter id="id14" name="check_obj_Tiny"
        value="TRUE"
        operator="equal" />
  <CheckerParameter id="id15" name="check_obj_Misaligned"
        value="TRUE""
        operator="equal" />
  <CheckerParameter id="id16" name="save_log_in_part"
        value="TRUE"
```

```
        operator="equal" />
</PLMXML>
```

The sample file in legacy XML format:

```
<?xml version=”1.0”?>
<?xml-stylesheet type=”text/xsl” href=”validation_rule.xsl”?
<Rule_Set
        name=”Test_rule_set_1”
        description=”Power Train Event 0 at Milestone 3"
        user="your_user_id" date="2005-12-25"
        xmlns="http://www.plmxml.org/Schemas/PLMXMLSchemaValidationRule"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.plmxml.org/Schemas/
PLMXMLSchemaValidationRule validation_rule.xsd"   >
<Rule>
        <Checker name=”mqc_check_dim_attach_solid”></Checker>
        <Mandated>no</Mandated>
        <Datasets>
            <type>UGMASTER</type>
            <type>UGPART</type>
        </Datasets>
        <Events class=”ITEMREVISION:owning_group” exclude=”yes”>
            <value>chassis.pe.company</value>
            <value>powertrain.pe.company</value>
        </Events>
</Rule>
<Rule>
        <Checker name=mqc_check_dim_attach_solid”></Checker>
        <Mandated>no</Mandated>
        <Datasets>
            <type>UGMASTER</type>
            <type>UGPART</type>
        </Datasets>
</Rule>
<Rule>
        <Checker name=”mqc_exam_geom_combo”></Checker>
        <Mandated>no</Mandated>
        <Datasets>
            <type>UGMASTER</type>
                <type>UGPART</type>
            <type>UGALTREP</type>
        </Datasets>
        <Events exclude=”no”>
            <value>Status*3</value>
            <value>Status_4</value>
        </Events>
        <Parameters>
            <Parameter operation=”=” title=”Check Tiny objects”
```

```
 value=”TRUE”></Parameter>
        </Parameters>
</Rule>
<Rule>
        <Checker name=”mqc_examine_geometry”></Checker>
        <Mandated>no</Mandated>
        <Datasets>
            <type>UGMASTER</type>
            <type>UGPART</type>
        </Datasets>
        <Events>
            <value>S*</value>
        </Events>
        <Parameters>
        <Parameter operation=”=” title=”Save Log in Part” value=
”TRUE”></Parameter>
</Rule>
</Rule_Set>
```

Create a validation rule file for NX RDDV results

You can use validation rules to verify the NX RDDV results.

1. Use **nx_check_requirement** for the checker name.

2. Specify the requirement item revision (item ID/revision) in the parameters section in the rule definition.

   Only one requirement item revision is allowed per validation rule. If a target part is subject to multiple requirements, multiple validation rules can be defined in the same file.

These rules in PLM XML format and in legacy format mean the **UGMaster** target parts must have a requirement-driven design result complying with the requirement revision 000001/A if the target item revision is under a certain owning group:

The rule file in PLM XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- GENERATED BY: PLM XML SDK 7.0.1.070 -->
PLMXML xmlns="http://www.plmxml.org/Schemas/PLMXMLSchema"
xmlns:rule="http://www.plmxml.org/Schemas/PLMXMLValidationRuleSchema"
schemaVersion="6" date="2009-11-17" time="14:21:25" author="fanj">
  <rule:ValidationRuleSet id="id1" name=" Example RDDV rules for single
requirement" ruleRefs="#id3">rule:ValidationRuleSet>
  <rule:ValidationRule id="id3" name="rule 1"
checker="nx_check_requirement"
 mandated="no" eventRefs="#id15" parameterRefs="#id12">
    <rule:Target id="id2" targetType="Dataset">
```

```
      <Item value="UGMASTER">Item value="UGMASTER">
    </rule:Target>
  </rule:ValidationRule>
  <CheckerParameter id="id12" name="Requirement" value="REQ-000001/A">
CheckerParameter id="id12" name="Requirement" value="REQ-000001/A">
  <rule:ValidationEvent id="id15" name="Event 6" eventClass=
"ITEMREVISION:owning_group" application="PLM">
    <rule:ValueList id="id22">
      <Item value="chassis.division.company">Item value=
"chassis.division.company">
      <Item value="powertrain">Item value="powertrain">
    </rule:ValueList>
  </rule:ValidationEvent>
</PLMXML>
```

The rule file in legacy XML format:

```
<?xml version=”1.0”?>
<?xml-stylesheet type=”text/xsl” href=”validation_rule.xsl”?
<Rule_Set
name=”Test_rule_set_1”
description=” Example RDDV rules for sinel requirement””
user="your_user_id" date="2005-12-25"
xmlns="http://www.plmxml.org/Schemas/PLMXMLSchemaValidationRule"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.plmxml.org/Schemas/
PLMXMLSchemaValidationRule validation_rule.xsd" >
<Rule>
  <Checker name="nx_check_requirement">Checker
name="nx_check_requirement">
  <Datasets>
    <type>UGMASTER<type>
  </Datasets>
  <Events exclude=”no”>
    <value>chassis.division.company</value>
    <value>powertrain</value>
  </Events>
  <Parameters>
    <Parameter operation=”=” title=”requirement”
      value=”0000001/A”>value=”0000001/A”>
  </Parameters>
</Rule>
</Rule_Set>
```

Define the following two rules within one file to ensure that the target parts have requirement-driven design results that comply with both the **000001/A** and **000002/A** requirement revisions:

Rule file in PLM XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- GENERATED BY: PLM XML SDK 7.0.1.070 -->
<PLMXML xmlns="http://www.plmxml.org/Schemas/PLMXMLSchema"
xmlns:rule="http://www.plmxml.org/Schemas/PLMXMLValidationRuleSchema"
schemaVersion="6" date="2009-11-17" time="14:21:25" author="fanj">
  <rule:ValidationRuleSet id="id1" name="Rules for RDDV"
 ruleRefs="#id3" #id7"></rule:ValidationRuleSet>
  <rule:ValidationRule id="id3" name="rule 1" checker=
"nx_check_requirement" mandated="no" parameterRefs="#id12">
    <rule:Target id="id2" targetType="Dataset">
      <Item value="UGMASTER"></Item>
    </rule:Target>
    </rule:ValidationRule>
  <rule:ValidationRule id="id7" name="rule 1" checker=
"nx_check_requirement" mandated="no" parameterRefs="#id20">
    <rule:Target id="id6" targetType="Dataset">
      <Item value="UGMASTER"></Item>
    </rule:Target>
    </rule:ValidationRule>
  <CheckerParameter id="id12" name="Requirement" value=
"REQ-000001/A"></CheckerParameter>
  <CheckerParameter id="id20" name="Requirement" value=
"REQ-000002/A"></CheckerParameter>
</PLMXML>
```

Rule file in legacy XML format:

```
<?xml version=”1.0”?>
<?xml-stylesheet type=”text/xsl” href=”validation_rule.xsl”?
<Rule_Set
        name=”Test_rule_set_1”
        description=”Example RDDV rules for multiple requirements"
        user="your_user_id" date="2005-12-25"
        xmlns="http://www.plmxml.org/Schemas/PLMXMLSchemaValidationRule"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.plmxml.org/Schemas/
PLMXMLSchemaValidationRule validation_rule.xsd"   >
<Rule>
     <Checker name=”nx_check_requirement”></Checker>
     <Mandated>no</Mandated>
     <Datasets>
        <type>UGMASTER</type>
     </Datasets>
     <Parameters>
        <Parameter operation=”=” title=”requirement” value=
”0000001/A”></Parameter>
     </Parameters>
</Rule>
<Rule>
```

```
      <Checker name=”nx_check_requirement”></Checker>
      <Mandated>no</Mandated>
      <Datasets>
         <type>UGMASTER</type>
      </Datasets>
      <Parameters>
         <Parameter operation=”=” title=”requirement” value=
”0000002/A”></Parameter>
      </Parameters>
</Rule>
</Rule_Set>
```