TEAMCENTER

MBSE Integration Gateway — Deployment

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

1. What is Teamcenter MBSE Integration Gateway?

Teamcenter MBSE Integration Gateway is an integration framework used to integrate modeling tools with Teamcenter. This framework is deployed on top of Teamcenter.

The integrations supported by Teamcenter MBSE Integration Gateway fit in the Model-Based Systems Engineering area.

Products such as a car, ship, or phone can be represented logically as a system model. If we consider a car as a system, components such as the engine, transmission, and brakes can be considered as subsystems. In Teamcenter, you can create a model of the system that depicts the different subsystems and the interactions between them. You can also create models of the requirements and functions. All these models created in Teamcenter are used by tools in different domains such as Failure Analysis, Functional Analysis, and Behavior Analysis. A system model is the core input to these tools. These tools generate simulation models based on the system models, and the results of the simulations are used in the downstream processes. All these tools must be integrated with Teamcenter. Teamcenter MBSE Integration Gateway provides a framework for this integration.

By integrating the modeling tools with Teamcenter, you can use them for model authoring and Teamcenter for model management. The integration helps leverage other Teamcenter benefits. You can:

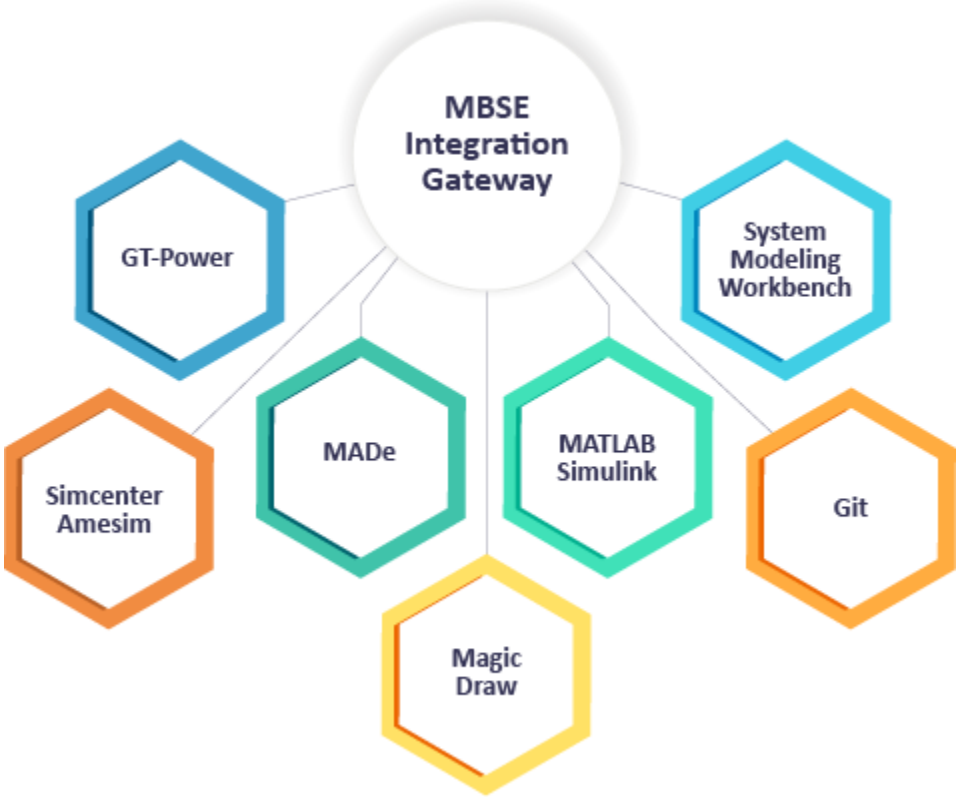
- Manage model data in Teamcenter in the context of the product data.
- Manage additional model data such as test results, reference data, and documentation in Teamcenter.
- Access the models saved in Teamcenter from the model authoring tools.
- Reuse models by organizing them in your enterprise model library, using the Teamcenter Classification application.

The Teamcenter MBSE Integration Gateway is a generic integration framework and can integrate with any behavior modeling tool and provides the following features, which are required for integrating any modeling tool with Teamcenter:

- Establishing connections with Teamcenter, including SSO
- Defining the data model and its deployment
- Defining the mapping of external tool artifacts with Teamcenter business objects
- Managing file uploads and downloads
- Providing the user interface to support Teamcenter operations

- Managing customization and business object extensibility
- Maintaining compatibility with multiple Teamcenter platforms

The following graphic shows a few of the available integrations.

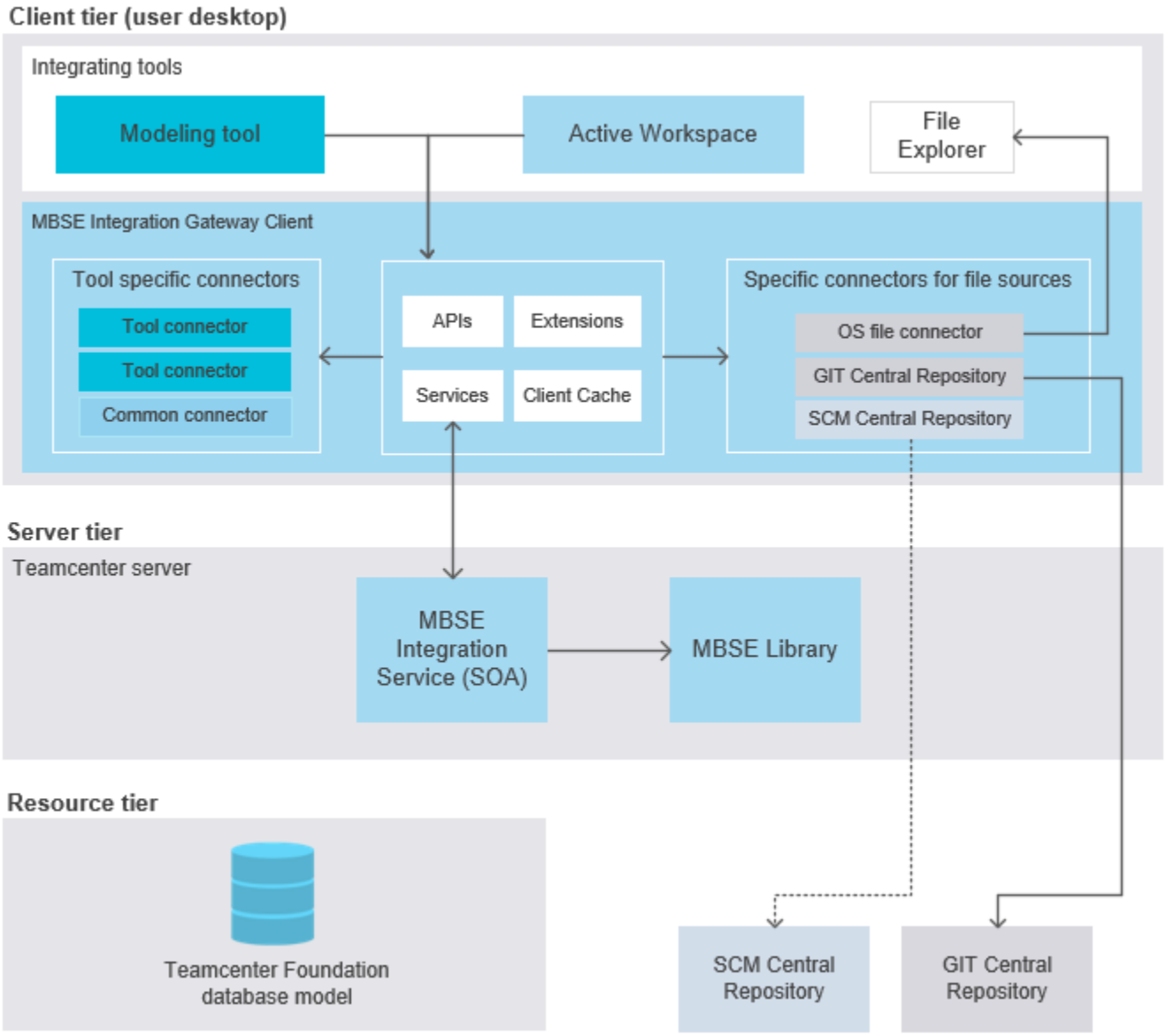


Where do I go from here?

Customizer	See Connecting to Teamcenter Using the MBSE Integration Gateway API for customizing the integration or creating new integrations.
Administrator	
Learn about the framework	See the sections on understanding the MBSE Integration Gateway framework and understanding the different integration modes.
Plan the deployment	Before you deploy the framework and your desired integration, plan the deployment considering factors such as the integration mode, modeling tool, and compatible versions of Teamcenter.
What are the scenarios, prerequisites, and overarching steps for deployment?	Refer to the checklist for deploying Teamcenter MBSE Integration Gateway.

2. Getting started with Teamcenter MBSE Integration Gateway

Understanding the Teamcenter MBSE Integration Gateway framework



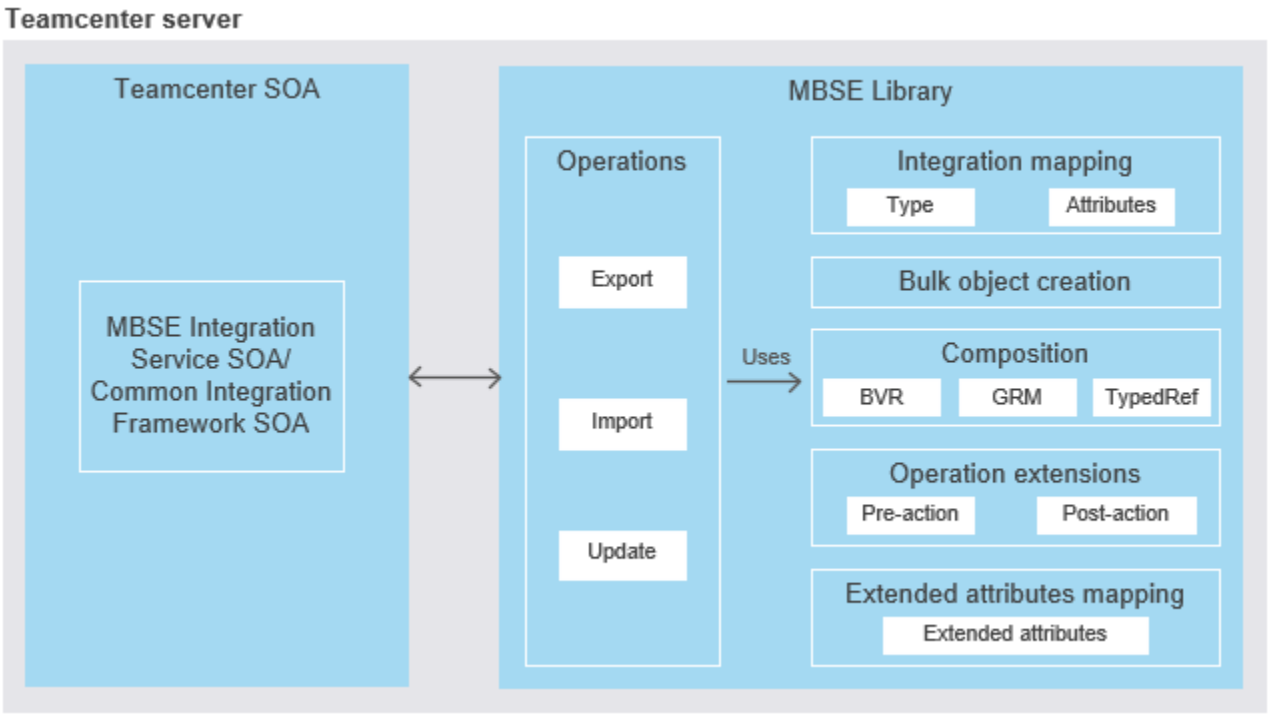
The MBSE Integration Gateway framework is the framework that supports the integration of modeling tools with Teamcenter. It consists of two main components:

- Server component

- Client component

Server component

The server component resides on the Teamcenter server and contains:



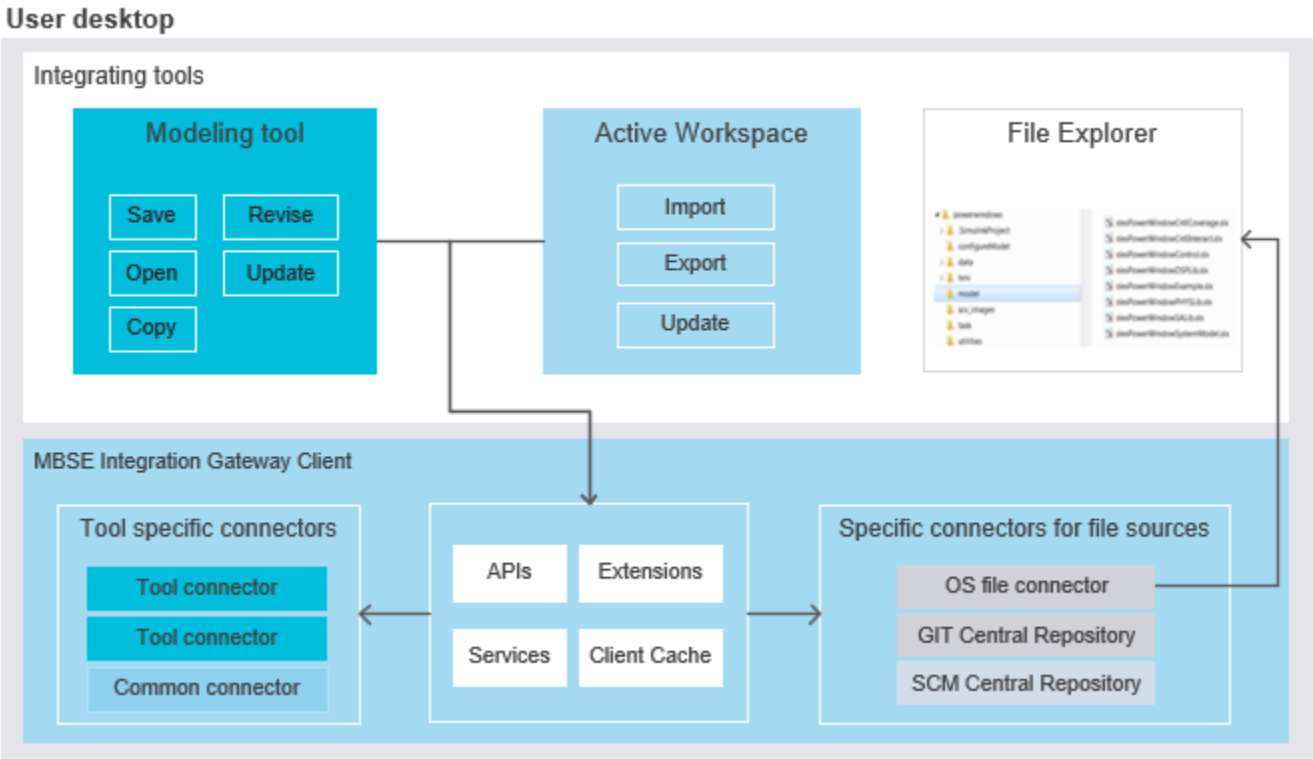
- The MBSE Integration Service. This service contains MBSE Integration Gateway SOAs also called the Common Integration Framework SOAs. It takes input from the MBSE Integration Gateway client component and processes it using the MBSE library components.

- The MBSE Integration library has the operations, extensions, and other services for processing the payload. You can access these services using the Common Integration Framework SOAs.

- When you install the integrations, the data models for the same are also installed in the Teamcenter database. If you have a custom integration, ensure that you create the data model or use the data model provided by MBSE Integration Gateway.

Client component

The client component is installed on the user desktop and consists of:



1. MBSE Integration Client: This component has MBSE Integration Gateway SOAs and other components such as the client cache. This component passes the JSON payload to the Teamcenter server.

2. Connectors: The connector exchanges data between the modeling tool and Teamcenter. The data that is exchanged is mapped using a file called the integration definition file. The following types of connectors are available:

    a. Tool specific connectors: These connectors are available for specific integrations and become available when you install the integrations on the client machine. The connectors are used to process tool specific data.

    b. Common connector: This connector is used for the common or generic integration.

    c. File specific connector: This connector is used for file-based operations.

3. Modeling tool plugins: You can add plugins in your modeling tool for initiating operations with Teamcenter. You must ensure that the data you generate is in JSON format so that it can be processed by the prod-mecheng; client.

4. Active Workspace: Integration commands are available in Active Workspace.

You can also develop your client components such as connectors or plugins using the MBSE Integration Gateway Toolkit. The Toolkit is available in the Teamcenter kit.

Understanding the different integration modes

Models authored in the modeling tool are saved in Teamcenter as Teamcenter business objects. You must configure each modeling tool to import model data and integrate it with Teamcenter. There are two modes of integration: specific connector-based integration and common connector-based integration.

Specific connector-based integration

In this integration mode, a tool-specific connector is used to import model data into Teamcenter. This connector reads the integration definition file to decide the type of data to be imported into Teamcenter. The following data is saved to Teamcenter based on the configuration.

- The model file corresponding to the model is associated to the model item revision as **IMAN_specification**.
- A snapshot of the model is saved as an image and associated to the model item revision as **TC_Attaches**.
- All the files that the model is dependent on are imported as multiple datasets with each dataset corresponding to one file. Each dataset is associated to the model revision as **Bhm0AssociatedData**.
- If a configuration is provided in the integration definition file of a modeling tool to import all the files from specified folders, all these files are imported as individual datasets and associated to the model item revision based on the configured relation.
- If a tool integration is configured to capture the model hierarchy and its components, all the configured components are saved in Teamcenter as separate business objects. These are associated to the model revision object either through the BOM View Revision or through the relation **Bhm0HasComponents**.
- If there are any model properties mapped to Teamcenter business object attributes in the integration definition file, the values of these model properties are stored in the respective attributes in the model object in Teamcenter.

Common connector-based integration

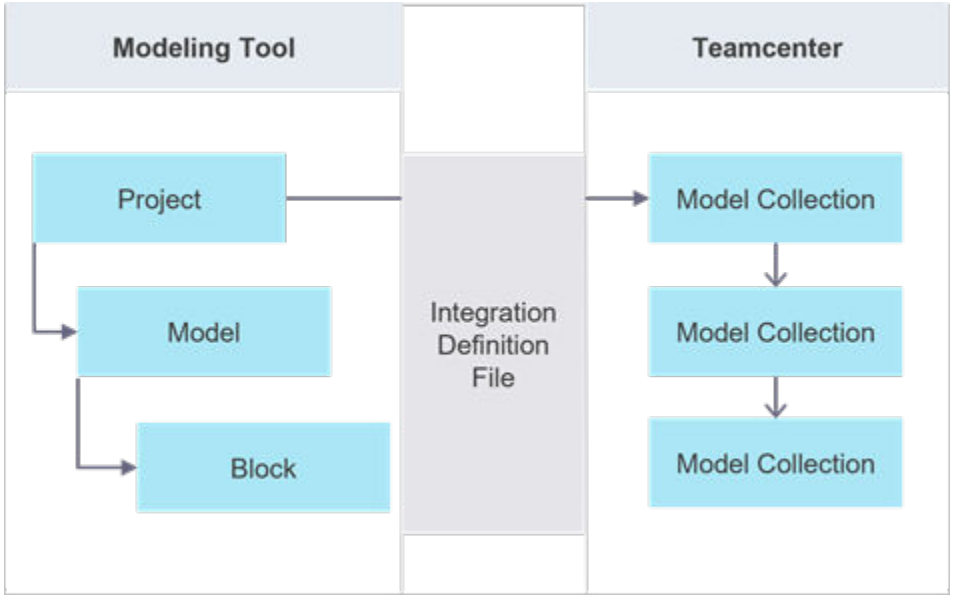
This integration mode provides limited integration of the modeling tool with Teamcenter. In this mode, all the model files in the operating system from a specific context folder are identified based on a configuration. These model files are then saved to Teamcenter as a model object. Only the model files are associated to these objects through the **IMAN_specification** relation. No other features of the connector-based integration mode are available.

Configuring integration operations using the integration definition file

Every integration of an external tool that is done using the MBSE Integration Gateway must have an integration definition file. This integration definition file contains all the configuration information needed by MBSE Integration Gateway to automatically perform various tasks without having to take user input for every operation that the Engineer is performing from the modeling tool.

This integration definition file is one per tool. It is stored in Teamcenter and associated to a dataset with name *APPLICATION_NAME_BHM_INT_DEF_FILE*, where *APPLICATION_NAME* is a unique identifier name for a tool that is being integrated.

The following figure shows two integration definition files. Each files shows the mapping of objects and relationships between the external tool and Teamcenter.



For more information about how to use the integration definition file, see *Integration definition file*.

3. Planning your Teamcenter MBSE Integration Gateway deployment

The MBSE Integration Gateway is a framework for integrating Teamcenter with modeling tools. To deploy this framework and your desired integration, plan your deployment based on the following guidelines:

- Choose the integration mode from the types of integration modes available:
  - Specific connector-based integration: This integration mode supports integration with specific modeling tools such as Simcenter Amesim, GTPower. A specific integration connector is available for the supported modeling tool.
  - Common connector-based integration (Enterprise Model Management): This integration mode provides limited integration of any modeling tool with Teamcenter. The following integrations are available with this mode:
    - Integration with file system data: All the model files in the operating system from a specific context folder are identified based on a configuration. You can import these model files into Teamcenter and manage your model lifecycle.
    - Integration with Git-based systems.
- Decide which modeling tool must be integrated with Teamcenter. Refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com to check which version of Teamcenter you need to install. Additionally, check if you must install only the server part of the MBSE Integration Gateway or you must also install both—the server part and the MBSE Integration Gateway client.
- Install or upgrade to the version of Teamcenter that supports the version of the modeling tool you want to integrate with Teamcenter.

After planning your deployment, refer to the **MBSE Integration Gateway deployment checklist** for more information.

4. Checklist for deploying Teamcenter MBSE Integration Gateway

Follow these steps to ensure successful deployment of your desired integration. Print the following scenario tables and follow the checklist from your chosen scenario:

- **Scenario 1: This is the first time you are deploying Teamcenter MBSE Integration Gateway**
- **Scenario 2: You have already deployed Teamcenter MBSE Integration Gateway and want to update it and install new features**

Scenario 1: This is the first time you are deploying Teamcenter MBSE Integration Gateway

	Step 1	**Prerequisites**
❒	Step 1.1	Decide which modeling tool must be integrated with Teamcenter. Refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com to check which version of Teamcenter you must install. You can also use the generic integration framework that works with all modeling tools.
❒	Step 1.2	Install or upgrade to the version of Teamcenter that supports the version of the modeling tool you want to integrate with Teamcenter.
❒	Step 2	Install the MBSE Integration Gateway framework and the server components **using TEM** or **Deployment Center**.
❒	Step 3	Perform the mandatory server configurations for your desired integration: • **Mandatory server configurations for Enterprise Model Management**
❒	Step 4	**Perform optional server configurations for your desired integration**
❒	Step 5	**Install the MBSE Integration Gateway client on a machine where your modeling tool is installed (if required)**
❒	Step 6	**Perform the optional client configurations for your desired integration**
❒	Step 7	Test the integration

		• **Test Enterprise Model Management (for generic integration functionality and Git integration functionality)**

Scenario 2: You have already deployed Teamcenter MBSE Integration Gateway and want to update it and install new features

	Step 1	**Prerequisites**
❒	Step 1.1	Refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com to check which version of Teamcenter supports the integration.
❒	Step 1.2	Update or upgrade to the version of Teamcenter that supports the version of the modeling tool you want to integrate with Teamcenter.
❒	Step 2	**Update MBSE Integration Gateway framework and the server components**
❒	Step 3	Check if you have performed the mandatory server configurations for your desired integration: • **Check if you have performed mandatory server configurations for Enterprise Model Management**
❒	Step 4	**Check if you must perform the optional server configurations for your desired integration**
❒	Step 5	**Update MBSE Integration Gateway client on a machine where your modeling tool is installed (if required)**
❒	Step 6	**Check if you need to perform the optional client configurations for your desired integration**
❒	Step 7	Test the integration • **Test Enterprise Model Management (for generic integration functionality and Git integration functionality)**

5. Scenario 1: First-time install of the MBSE Integration Gateway

Step 1: Perform the prerequisite steps

- Decide which modeling tool must be integrated with Teamcenter. Refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com to check which version of Teamcenter you must install. Also check if you need to install just the server part of the MBSE Integration Gateway or you also need to install the MBSE Integration Gateway client.

- Install or upgrade to the version of Teamcenter that supports the version of the modeling tool you want to integrate with Teamcenter.

After performing this step, perform **Step 2: Install Teamcenter MBSE Integration Gateway framework and server features** or refer to the checklist **Scenario 1: This is the first time you are deploying Teamcenter MBSE Integration Gateway** for next steps.

Step 2: Install Teamcenter MBSE Integration Gateway framework and server features using TEM

This topic describes how to install the MBSE Integration Gateway framework and the server components required for your integration.

Ensure that you have **performed the pre-requisites steps**. You can choose from the following installation options:

1. Launch TEM for the corporate server to which the MBSE Integration Gateway features are to be added.

2. In the **Install/Update Options** panel, click **Install**.

3. In the **Media Locations** panel, specify the:

    - Location of the base Teamcenter installation in the **Original Medial Location** box.

    - Location of the Teamcenter patch install file in the **Update Location** list.

    - Location of the Active Workspace install file in the **Update Location** list.

4. Click **Next** until you reach the **Features** panel.

5. Select from the any of the following features depending on the integration you require:

Functionality required	Feature to select
Install Teamcenter Polarion Direct Integration	**Extensions→ MBSE Integrations→ Server→ Teamcenter Polarion Direct Integration**
Install Safety Architect Integration	**Extensions→ MBSE Integrations→ Server→ Safety Architect Integration**
Install Capital Marine Integration	**Extensions→ MBSE Integrations→ Server→ Capital Marine Integration**
Install MADe Integration	**Extensions→ MBSE Integrations→ Server→ MADe Integration**
Install Software Management	**Extensions→ MBSE Integrations→ Server→ Software Management**
Install IBM Rhapsody Integration	**Extensions→ MBSE Integrations→ Server→ IBM Rhapsody Integration**
Install Magic Draw Integration	**Extensions→ MBSE Integrations→ Server→ Magic Draw Integration**
Install System Modeling Workbench Integration	**Extensions→ MBSE Integrations→ Server→ System Modeling Workbench Integration**

6. Click **Next** until you reach the **Confirmation** panel.

   In the **Confirmation** panel, click **Start**.

7. When the installation is complete, click **Close**.

After you have installed the server features, you can configure the integration features on the Teamcenter server. Choose from the following configuration tasks:

- **Configuring Enterprise Model Management**
- **Performing optional server configurations for Teamcenter MBSE Integration Gateway**

You can also review the checklist **Scenario 1: This is the first time you are deploying Teamcenter MBSE Integration Gateway**.

Step 2: Install Teamcenter MBSE Integration Gateway framework and server features using Deployment Center

This topic describes how to install the MBSE Integration Gateway framework and the server components required for your integration.

Prerequisites

Ensure that you have **performed the prerequisites steps**.

Procedure

1. Log on to Deployment Center and select the environment to which you want to add MBSE Integration Gateway.

2. Go to the **Applications** task. Click **Add or Remove Selected Applications**.

3. In the **Available Applications** panel, use the web browser search to find the following applications:

Functionality required	Application to select
Install Capital Marine Integration	**Capital Marine Integration**
Install IBM Rhapsody Integration	**Integration for IBM Rhapsody**
Install MADe Integration	**MADe Integration**
Install Magic Draw Integration	**MagicDraw Integration**
	Real-time co-authoring
	Requirements Management - Quality Module
Install Safety Architect Integration	**Integration for Safety Architect**
Install Software Management	**Software Management**
Install System Modeling Workbench Integration	**Integration for System Modeling Workbench**
Install Teamcenter Polarion Direct Integration	**Teamcenter Polarion Direct Integration**

   Select the application you need, and then click **Update Selected Applications**.

   Deployment Center automatically selects any additional dependent applications.

4. Go to the **Components** task.

5. In the **Selected Components** list, note any remaining components whose configuration status is not **100%**. Select each incomplete component, enter required parameters, and save component settings until all components in the environment show a configuration status of **100%**.

   When all components are fully configured, the **Deploy** task is enabled.

6. Go to the **Deploy** task. Click **Generate Install Scripts** to generate deployment scripts you will use to update affected machines.

   When script generation is complete, note any special instructions in the **Deploy Instructions** panel.

7. Locate deployment scripts, copy each script to its target machine, and then run each script on its target machine.

   For more information about running deployment scripts, see Deployment Center — Usage.

Postrequisites

After you have installed the server features, you can configure the integration features on the Teamcenter server. Choose from the following configuration tasks:

- **Configuring Enterprise Model Management**
- **Performing optional server configurations for Teamcenter MBSE Integration Gateway**

You can also review the checklist **Scenario 1: This is the first time you are deploying Teamcenter MBSE Integration Gateway**.

Step 3: Perform the mandatory server configurations for your desired integration

Configuring Enterprise Model Management

Enterprise Model Management configuration options

You can perform the following configurations for Git and file system integrations:

- **Choose how model collection objects are treated—as BVR structure or Workspace objects**
- **Enable the Git or file system integrations**
- **Enable image-preview display for models and model collections**
- **Configure additional properties for import**
- **Remove Git credentials when uninstalling MBSE Integration Gateway**

For the next steps following these configurations, refer to the **MBSE Integration Gateway deployment checklist**.

Choose whether model collection objects are treated as BVR structures or Workspace objects

Enterprise Model Management allows you to treat model collection objects as either Bom View Revision (BVR) structures or as Workspace objects.

BVR structures make revisions of model collections during each update, while Workspace objects create baselines. The BVR structure allows you to apply Teamcenter configurations to a model collection.

BVR structure

If you choose the BVR structure mode:

Update the *PROJECT_BHMIntegrationDefinition.xml* file. This file is the named reference in the **PROJECT_BHM_INT_DEF_FILE** dataset.

Add the following entry if it is not there:

```
  <ConnectorClass
fullName="com.teamcenter.behaviormodeling.filesandfolderint.FilesAndFolde
rsConnector" jarFilePath="" />
```

Choose Workspace objects

If you choose the Workspace objects mode:

Update the *PROJECT_BHMIntegrationDefinition.xml* file. This file is the named reference in the **PROJECT_BHM_INT_DEF_FILE** dataset.

Remove or comment out the **ConnectorClass** entry, which appears as follows:

```
  <ConnectorClass
fullName="com.teamcenter.behaviormodeling.filesandfolderint.FilesAndFolde
rsConnector" jarFilePath="" />
```

After completing this configuration, you can:

- Check if you need to perform any other **Git and file system configurations**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Configuring Git and file system integrations

Depending on the integration you want, choose from the following options:

- **Configure Git integration**
- **Configure file system integration**

Configuring Git integration

- Update the *PROJECT_BHMIntegrationDefinition.xml* file. This file is the named reference in the **PROJECT_BHM_INT_DEF_FILE** dataset.

Check if the XML file contains the following entry:

```
<!-- To specify SCM Tool Connector -->
<SCMToolConnector connectorType="GIT"
connectorClass="com.teamcenter.scmadapter.gitadapter.GitScmAdapter"
cloneInTC="true" toolName="gitlab" baseUrl="http://plgit/gitlab/"
localRepoPath="C:/temp/"/>
```

If the entry is not present, add an entry.

Entry name	Description
cloneInTC	If this is set to **true**, the Git files are also copied along with the metadata. If set to **false**, only the metadata is copied.
toolName	Specifies the name of the Git tool, such as GitHub or GitLab.
baseUrl	Specifies the base URL of the Git tool.
localRepoPath	Specifies the path where the Git files are temporarily downloaded before the files are imported into Teamcenter.

- If you are using the MATLAB connector, update the *MATLAB_BHMIntegrationDefinition.xml* file. This file is the named reference in the **MATLAB_BHM_INT_DEF_FILE** dataset.

  Update the entry from:

  ```
  <ConnectorClass
  fullName="com.teamcenter.behaviormodeling.matlabint.MatlabConnector"
  jarFilePath="" />
  ```

  To:

  ```
  <ConnectorClass
  fullName="com.teamcenter.behaviormodeling.commonclient.defaultconnector
  .CommonConnector" jarFilePath="" />
  ```

Configuring file system integration

Update the *PROJECT_BHMIntegrationDefinition.xml* file. This file is the named reference in the **PROJECT_BHM_INT_DEF_FILE** dataset.

If the following entry is present, comment it out.

```
<!-- To specify SCM Tool Connector
<SCMToolConnector connectorType="GIT"
connectorClass="com.teamcenter.scmadapter.gitadapter.GitScmAdapter"
cloneInTC="true" toolName="gitlab" baseUrl="http://plgit/gitlab/"
localRepoPath="C:/temp/"/> -->
```

After completing this configuration, you can:

- Check if you need to perform any other **Git and file system configurations**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable image preview for models and model collections

When using the common connector-based integration, model image previews are not available. To enable image previews, do the following:

1. Generate the image file using a custom logic implemented as extensions on existing integration operations, or generate it manually.

2. Copy the image files to a folder. The location of the folder can be anywhere on the file system, or it can be within the root folder of the model collection. The path to this folder must be specified in the integration definition file.

   - For a model collection, this location must be specified in the *PROJECT_integration_definition_file.xml* file, while for a specific tool integration, the location should be specified in the respective integration definition file. For example, for MATLAB Simulink, the location must be specified in the *MATLAB_integration_definition_file.xml* file.
   - The path can be an absolute path if it is outside the root folder of the model collection.
   - If the absolute path is not specified, the name of the folder is used to create a folder within the root folder of the model collection.
   - Specify the relation type in the integration definition file as either **IMAN_Rendering** or **TC_Attaches**. The image files are associated to the model or model collection with this relation.

   Example:

   Example of configuration for a model collection in the *PROJECT_integration_definition_file.xml* file:

```
<ObjectMapping type="Project" tctype="Bhv0ModelCollection">
    <BHMElement type="RootModel" tctype="Bhv0ModelCollection">
        <AttributeMappings >
            <!-- <AttributeMapping name="DESCRIPTION" tcattr="object_desc"/> -->
        </AttributeMappings >
    </BHMElement>
    <OrganizationData>
        <Folder name="Viewables" tcRelation="IMAN_Rendering"/>
    </OrganizationData>
</ObjectMapping>
```

Example:

Example of configuration for Simulink models in the *MATLAB_integration_definition_file.xml* file:

```
<ObjectMapping type="Model" behaviorType="BVR" tctype="Bhm0BehaviorModl">
    <BHMElement type="RootModel" tctype="Bhm0BehaviorModl">
        <AttributeMappings >
            <RevisionAttributeMapping>
                <AttributeMapping name="TOOL_VERSION" tcattr="bhm0toolVersion"/>
            </RevisionAttributeMapping>
        </AttributeMappings >
    </BHMElement>

    <OrganizationData>
        <Folder name="image" tcRelation="IMAN_Rendering"/>
    </OrganizationData>
</ObjectMapping>
```

3. If the image folder is in the root folder and you do not want this information in Teamcenter under model collection, add the folder to the exclusion list.

The model management client reads the integration definition file and imports the file as datasets. These imported datasets are then associated with the models or model collection using the specified relation.

After completing this configuration, you can:

- Check if you need to perform any other **Git and file system configurations**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Configure additional properties for import

When you import models from Git to Teamcenter, you can import additional properties. On successful import, the properties are associated with the Teamcenter model collection. For the import to work, the following conditions must be met:

- The properties must be defined in the *PROJECT_BHMintegrationDefinition.xml* file.

- The Teamcenter property that you want to associate with must exist.

To import properties:

1. Create a property file that contains the properties that you want to import. The properties in the property file must have a name-value pair.

   Example:

   Create a property file named *properties_input.txt*, and add the following property in the name-value format as follows:

   ```
   //name = DESCRIPTION, value=This data was imported from GitHub.
   DESCRIPTION This data was imported from GitHub.
   ```

2. In the *PROJECT_BHMintegrationDefinition.xml* file, add the property you defined in the property file, and specify the Teamcenter property it maps to.

   Example:

   ```
   <ObjectMapping type="Project" tctype="Fnd0Branch">
     <BHMElement type="RootModel" tctype="Fnd0Branch">
       <AttributeMappings >
         <AttributeMapping name="DESCRIPTION" tcattr="object_desc"/>
       </AttributeMappings >
     </BHMElement>
   </ObjectMapping>
   ```

   In this example, the element is **AttributeMapping**, and it has the **name** argument with the value **Description**. The value of the **name** argument must match the property in the property file created in the previous step.

   In this example, the property maps to a Teamcenter property named **object_desc**.

   Make sure that this Teamcenter property exists because if not, the import action ignores this entry.

3. Check if the additional properties are imported and associated to the model collection by importing Git data into Teamcenter.

After completing this configuration, you can:

- Check if you need to perform any other **Git and file system configurations**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Remove Git credentials when uninstalling MBSE Integration Gateway

The Git integration uses the Windows Credential Manager to store the Git login credentials that allows you to automatically log on to Git. When you uninstall MBSE Integration Gateway or model management functionality ensure that you remove the Git credentials as follows:

1. Open the **Windows Credential Manager** from the Windows Start search.

2. Open the **Windows Credentials** tab.

3. Delete the git login credentials related to **GITSCMADAPTER** from the **Generic Credentials** section.

After completing this configuration, you can:

- Check if you need to perform any other **Git and file system configurations**.

- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Step 4: Performing optional server configurations for Teamcenter MBSE Integration Gateway

List of optional server-side configurations for Teamcenter MBSE Integration Gateway

Perform the following optional server-side configurations for Teamcenter MBSE Integration Gateway:

Configuration	Description
Mapping data using the integration definition file	To customize how the data is mapped between Teamcenter and the modeling tool
Configure the integration definition file for different users	To customize the integration behavior for different users or groups
Set up common connector-based integration	To enable integration with the tool of your choice when you are using the generic or common connector-based integration
Enable the Active Workspace Open in Tool command	To enable the Open in Tool command in Active Workspace that allows you to open models from Active Workspace in the respective modeling tools
Configure a verification request	To enable the use of a *Verification Request* with MBSE Integration Gateway
Classify models	To set up the classification of models in Teamcenter
Enable the sample classification library for model management	To enable the sample classification library to manage your models in Teamcenter
Configure the data to be excluded during import	To choose which data must be excluded during import
Create behavior modeling objects in Teamcenter	To create custom behavior modeling objects in Teamcenter
Configure how long object and occurrence names are handled	To handle object names greater than 128 characters and occurrence names greater than 100 characters.

After performing these configurations, to see the next steps, refer to the **MBSE Integration Gateway deployment checklist**.

Mapping the data using the integration definition file

Integration definition file

Every integration of an external tool that is done using the MBSE Integration Gateway must have an integration definition file. Configuration of the integration definition file is mandatory for the following SOAs:

- **exportCollection()**
- **importCollection()**
- **updateCollection()**

The integration definition file provides a way to configure mapping of tool artifacts and its metadata to Teamcenter business objects and properties. This integration definition file is one per tool. It is stored in a dataset with name *APPLICATION_NAME_BHM_INT_DEF_FILE*, where *APPLICATION_NAME* is a unique identifier name for a tool that is being integrated. The dataset must have administrative control.

Definition of unique application name

Every tool that gets integrated with Teamcenter is identified uniquely by the integration gateway. This information is provided by the integrators in the integration definition file as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<BHMIntegration xmlns="http://www.plmxml.org/Schemas/bhm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.plmxml.org/Schemas/bhm"
applicationName="ECAD">
```

When various operations are performed, the integrator has to provide the value from the attribute **applicationName** as input.

Definition of the connector class

The integration gateway is executed in two modes:

- Push: In this mode, the integrating tools extract the model data and pushes this extracted data to the modeling gateway using the published APIs. This is typically done when the tool adapter or connector is an integral part of the tool.

- Pull: In this mode the integration gateway pulls the model data by invoking APIs published by the tool specific adapter or connector. The information related to the tool specific adapter or connector to be used is provided in the integration definition file. The pull mechanism is useful when bulk operations are to be performed for importing the modeling data from various modeling tools within a given modeling project.

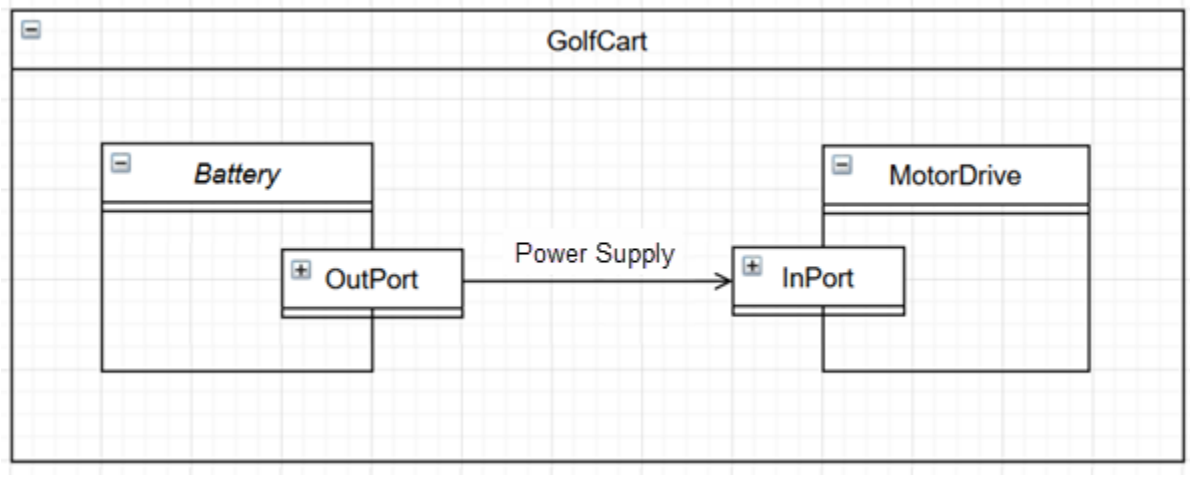
The details of the connector are given in the integration definition file as follows:

```
<ConnectorClass
fullName="com.teamcenter.behaviormodeling.matlabint.MatlabConnector"
jarFilePath="" />
```

Definition of the containers

Every modeling tool has at least two types of artifacts: a parent which acts like a container and a child which acts like a component within the container.

The container, when the model data is exported to Teamcenter from the modeling tool, is always represented in Teamcenter as an instance of some business object type. It can be a classic business object like Item or an object like dataset that holds the files or any generic workspace or POM object. In the modeling tool some containers may have children which are termed as components. When the model data is imported in Teamcenter these components are translated into instances of relationship. These relationships could be GRM, BVR or just a typed reference property on the container or a back pointer on the child container. Thus, container is considered as some business object type to be created in Teamcenter and component as some relationship. The integration definition file facilitates defining various containers from the modeling tool and its mapping to corresponding Teamcenter business object type.



In the GolfCart example GolfCart, Battery, MotorDrive, OutPort, Inport, PowerSupply all are containers.

The following code sample is a definition of a container. For each type, you must define **ObjectMapping**. Mapping for a parent type is not automatically applicable to its sub-types.

GolfCart, Battery, MotorDrive all are treated as containers by the Integration Gateway. If they are of type Model in the tool, they are mapped to the Teamcenter business object Item as follows:

```
<ObjectMapping type="Model" behaviorType="BVR" tctype="Item">
```

If the InPort and OutPort containers are of type port in the modeling tool they are mapped to Teamcenter business object GeneralDesignElement.

```
<ObjectMapping type="port" behaviorType="MIXED"
tctype="GeneralDesignElement">
```

If physical model file of GolfCart that is generated by the tool is *GolfCart.zip* then it can be stored in a Teamcenter business object of type **Zip** and is mapped as follows:

```
<ObjectMapping type="Design" behaviorType="MIXED" tctype="Zip">
```

where:

- **ObjectMapping** corresponds to the definition of a container which maps type of tool artifact to Teamcenter business object type.

ObjectMapping attributes:

- **type** is the type of container or artifact in the modeling tool.
- **tcType** is the type of business object in Teamcenter representing the container.
- **behaviorType** is the type of association between an object corresponding to the **ObjectMapping** and the object corresponding to the **BHMElement**. It can be overridden at the component level as explained in the component definition section. It may have one of the following values:
  - **BVR** If all the components are BVR occurrences.
  - **GRM** If all the components are associated with GRM relations.
  - **GBVR** If all the components are of **GBVR** type.
  - **REF** If all the components are of **REF** type.
  - **MIXED** If components are associated with more than one type of **behaviorType** such as **BVR**, **GRM**, **REF**, or **GBVR**.

Definition of all the components

Within a modeling tool a container can contain various types of components.

Each of these components are usages of some container or artifact from the tool. A component represents an association between two containers. In the integration definition file **BHMElement** is used to configure this association.

```
<ObjectMapping type="Model" behaviorType="MIXED"
tctype="Bhm0BehaviorModlItem">
  <BHMElement type="Model" tctype="Item" behaviorType="BVR">

  <BHMElement type="Design" tctype="Zip" behaviorType="GRM"
reltype="IMAN_specification" reftype ="Design" isPrimary=”false”
isPrimaryAnchor=”false” isSecondaryAnchor=”false”>

  <BHMElement type="port" tctype="GeneralDesignElement"
```

```
behaviorType="GBVR">
</ObjectMapping>
```

where:

- **BHMElement** configures the component, that is the association between two containers or **ObjectMapping**.
- **type** is the component type from the modeling tool.
- **behaviorType** is the association definition with its components.

  It may have one of the following values:

  - **BVR** If there is a parent-child (occurrence) association between the object corresponding to the **ObjectMapping** and the object corresponding to the **BHMElement**. The object corresponding to the **ObjectMapping** is parent and the object corresponding to the **BHMElement** is a child. When **behaviorType** is **BVR**, value of **relType** is not required because a Teamcenter object of type **PSOccurrence** is automatically created to associate the parent and the child object.

    For example, GolfCart and Battery containers of type Model are represented by the Teamcenter business object item. These containers have a parent-child association where GolfCart is a parent and Battery is a child.

  - **GRM** If there is a GRM relation between the object corresponding to the **ObjectMapping** and the object corresponding to the **BHMElement**. The relation object is a subtype of **ImanRelation**. Any relation object has one primary object and one secondary object. By default, the container object corresponding to the **ObjectMapping** is the primary object and the container object corresponding to the **BHMElement** is the secondary. This can be changed by configuring the **isPrimary** value.

    By default, the relation is created between the revisions of the objects. You can override this behavior by configuring the **isPrimaryAnchor** and **isSecondaryAnchor** values.

  - **GBVR** If the object corresponding to the **BHMElement** is a port object.

  - **REF** If the object corresponding to the **BHMElement** is associated with a container object corresponding to **ObjectMapping** through a referenced property.

- **relType** defines the relation between the container corresponding to **ObjectMapping** and the container corresponding to **BHMElement**. It is a subtype of the Teamcenter business object **ImanRelation**.
- **isPrimary** This is an optional input. It is valid only if **behaviorType** is **GRM**. Every GRM relation has one primary object and one secondary object. If value of **isPrimary** is **true**, then the object mapped in **BHMElement** is primary. If value of **isPrimary** is **false**, then the object mapped in **ObjectMapping** is primary.

- **isPrimaryAnchor** This is an optional input. It is valid only if **behaviorType** is **GRM**. Every GRM relation has one primary object and one secondary object. If value of **isPrimaryAnchor** is **true**, then item of the primary object is associated with the secondary object. If value of **isPrimaryAnchor** is **false** or not set, then item revision of the primary object is associated with the secondary object.

- **isSecondaryAnchor** This is an optional input. It is valid only if **behaviorType** is **GRM**. Every GRM relation has one primary object and one secondary object. If value of **isSecondaryAnchor** is **true**, then item of the secondary object is associated with the primary object. If value of **isSecondaryAnchor** is **false** or not set, then item revision of the secondary object is associated with the primary object.

  The following are possible combinations of **isPrimaryAnchor** and **isSecondaryAnchor** values and its result on the primary object and secondary object while creating a GRM relation:

isPrimaryAnchor	isSecondaryAnchor	Container object corresponding to ObjectMapping	Container object corresponding to BHMElement
false/empty/absent	false/empty/absent	ItemRevision	ItemRevision
false/empty/absent	true	ItemRevision	Item
true	false/empty/absent	Item	ItemRevision
true	true	Item	Item

Definition of attribute mapping

Attribute mapping provides a way to map tool side metadata to properties on Teamcenter business objects. The integration definition file provides attribute mapping capability at the container as well as component level which translates to attributes on the Teamcenter business object as well as on the **Iman_relation** or **Occurrence**. For example, when an integrating tool artifact is represented by a Teamcenter business object such as item, then the artifact name can be stored in the **object_name** property on item. The integration definition file provides a way to configure this mapping of artifact name to the object_name property of the item.

When the **AttributeMapping** is inside the **BHMElement** and if the **type** value of **BHMElement** is **RootModel** then the mapped attributes are on the Container object else they are on the association object-**GRM** relation or **BVR** occurrence.

```
 <AttributeMappings>
      <AttributeMapping name="Description" tcattr="object_desc"/>
      <AttributeMapping name="cifContainerIdentifier"
tcattr="Cif0ToolSpecificIntInfo::cif0ToolProperties"/>
      <RevisionAttributeMapping>
        <AttributeMapping name="Description" tcattr="object_desc"/>
                        <AttributeMapping name="FormDescription"
tcattr="Form::object_desc"/>
                        <AttributeMapping name="cifContainerIdentifier"
```

```
tcattr="Cif0ToolSpecificIntInfo::cif0ToolProperties"/>
                        <AttributeMapping name="ExtraMetaData1"
tcattr="Cif0ToolSpecificIntInfo::cif0ToolProperties"/>
      </RevisionAttributeMapping>
    </AttributeMappings>
```

where:

- **name** is the name of model property from the modeling tool.
- **tcattr** is the name of the attribute name on the corresponding mapped Teamcenter business object.

  Property on Form object is prefixed with **Form::**. This functionality supported only by Common Integration Framework SOAs. If there is some extra information that needs to be stored but cannot be mapped to any of the properties on a Teamcenter object it is configured as **Cif0ToolSpecificIntInfo::cif0ToolProperties**. This functionality supported only by Common Integration Framework SOAs.

  Mapping **cifContainerType** as **Cif0ToolSpecificIntInfo::cif0ToolProperties** is mandatory if multiple **ObjectMapping::type** are mapped to single **ObjectMapping::tctype**.

Integrators can map multiple attributes. The attribute mapping is supported at the Item, ItemRevision and Form level for an item type of object.

Definition of file mappings

In most of the integrations there are many different types of files that need to be imported and associated to the container objects in Teamcenter. Integration definition file supports mapping the file extension type to different dataset types and named reference in Teamcenter.

```
<FileMapping>
    <FileMap fileExt="txt" tcDatasetType="Text"
tcNameReferencedType="Text" />
    <FileMap fileExt="png" tcDatasetType="Image"
tcNameReferencedType="Image" />
    <FileMap fileExt="html" tcDatasetType="HTML"
tcNameReferencedType="HTML" />
    <FileMap fileExt="jpg" tcDatasetType="JPEG"
tcNameReferencedType="JPEG_Reference" />
    <FileMap fileExt="zip" tcDatasetType="Zip"
tcNameReferencedType="ZIPFILE" />
    <FileMap fileExt="tif" tcDatasetType="TIF"
tcNameReferencedType="TIF_Reference" />
    <FileMap fileExt="gif" tcDatasetType="GIF"
tcNameReferencedType="GIF_Reference" />
    <FileMap fileExt="*" tcDatasetType="MISC"
```

```
tcNameReferencedType="MISC_TEXT" />
</FileMapping>
```

where:

- **fileExt** is the file extension.
- **tcDatasetType** is the type of dataset that shall contain this file.
- **tcNamedReferenceType** is the type of named reference within a dataset that should be used to associate the file to the dataset.

Definition of standard folders

Many times, a system is managed as modeling project that contains various different models as a part of that project. Each project has standard folder structure and has standard output or generated files residing in some standard folders. Many times, when the models are saved to Teamcenter this derived information has to be saved and associated to the model in Teamcenter. This configuration provides a facility for the integrators to specify such standard folders so that the integration gateway automatically imports all the files in these standard folders in Teamcenter and associates it to the model object.

```
<OrganizationData>
  <Folder name="MODELFOLDER" tcRelation="TC_References"/>
</OrganizationData>
```

where:

- **name** is the name of the standard folder. **MODELFOLDER** is keyword indicating the model in which model file is residing.
- **tcRelation** specifies the type of **IMAN_Relation** to be used to associate the files from this folder to the model object

Definition of extension points

The integration operations exhibit a certain behavior. This behavior may or may not cover all the use cases of the integration. Hence the integration gateway supports extending these operations on the client side by providing extension points on all the supported operations of save, open and update. The extension points are of three types:

- **PRE-CONDITION**: Executed before the base operation begins. If this extension fails, the operation is fails.
- **PRE-ACTION**: Executed after the pre-condition but before the base operation. If this extension fails, the error is logged in log file but the operation proceeds.

- **POST-ACTION**: Executed after the base operation execution is complete. If this extension fails, the error is logged in log file but the operation proceeds.

The extension can be implemented as JAVA code, batch file or in tool native language.

```
<Extensions>
    <Extension operationName="SAVE" extensionPoint="PRE_CONDITION" >
    <Impl type="SCRIPT" location="" name="SAVE_PRE_CONDITION" />
    </Extension>
    <Extension operationName="SAVE" extensionPoint="PRE_ACTION" >
    <Impl type="BATCH" location="D:\temp" name="RunMe.bat" />
    </Extension>
    <Extension operationName="SAVE" extensionPoint="POST_ACTION" >
    <Impl type="JAVA" location=""
name="com.teamcenter.matlabcustom.SaveCustom" />
    </Extension>
</Extensions>
```

where:

- **operationName** is the name of the operation for which the extension point is being defined. It can be **SAVE**, **OPEN**, or **SAVEAS**.
- **extensionPoint** is type of extension point such as **PRE_CONDITION**, **PRE_ACTION**, or **POST_ACTION**.
- **type** is the type of implementation.
- **location** is the location on file system where the jar, script, or batch file that implements this extension resides.
- **name** is the name of the class or file that actually implements the extension.

Map MADe data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of MADe by updating the MADe integration definition file. The file is named *MADe_BHM_INT_DEF_FILE*, and its format is similar to the format of the **integration definition file**.

> Note:
>
> If you have defined custom behavior modeling objects in Teamcenter and the default values of those objects are mandatory, you must map the attributes to MADe objects using the integration definition file.

To update the integration definition file:

1. In Teamcenter, search for **MADe_BHM_INT_DEF_FILE**.
2. Check out the dataset and download the named reference and update the file.
3. Check in the dataset to Teamcenter.
4. Log on to the MADe integration client to use the updated configuration.

Map MagicDraw data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of MagicDraw by updating the MagicDraw integration definition file. The file is named *MagicDraw_BHM_INT_DEF_FILE*, and its format is similar to the format of the **integration definition file**.

> Note:
>
> If you have defined custom behavior modeling objects in Teamcenter and the default values of those objects are mandatory, you must map the attributes to MagicDraw objects using the integration definition file.

To update the integration definition file:

1. In Teamcenter, search for **MagicDraw_BHM_INT_DEF_FILE**.
2. Check out the dataset and download the named reference and update the file.
3. Check in the dataset to Teamcenter.
4. Log on to the MagicDraw integration client to use the updated configuration.

Map System Modeling Workbench data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of System Modeling Workbench by updating the SYSML integration definition file. The file is named *SYSML_BHMIntegrationDefinition*, and its format is similar to the format of the **integration definition file**.

> Note:
>
> If you have defined custom behavior modeling objects in Teamcenter and the default values of these objects are mandatory, you must map the attributes to System Modeling Workbench objects using the integration definition file.

To update the integration definition file:

1. In Teamcenter, search for **SYSML_BHMIntegrationDefinition**.

2. Check out the dataset and download the named reference and update the file.

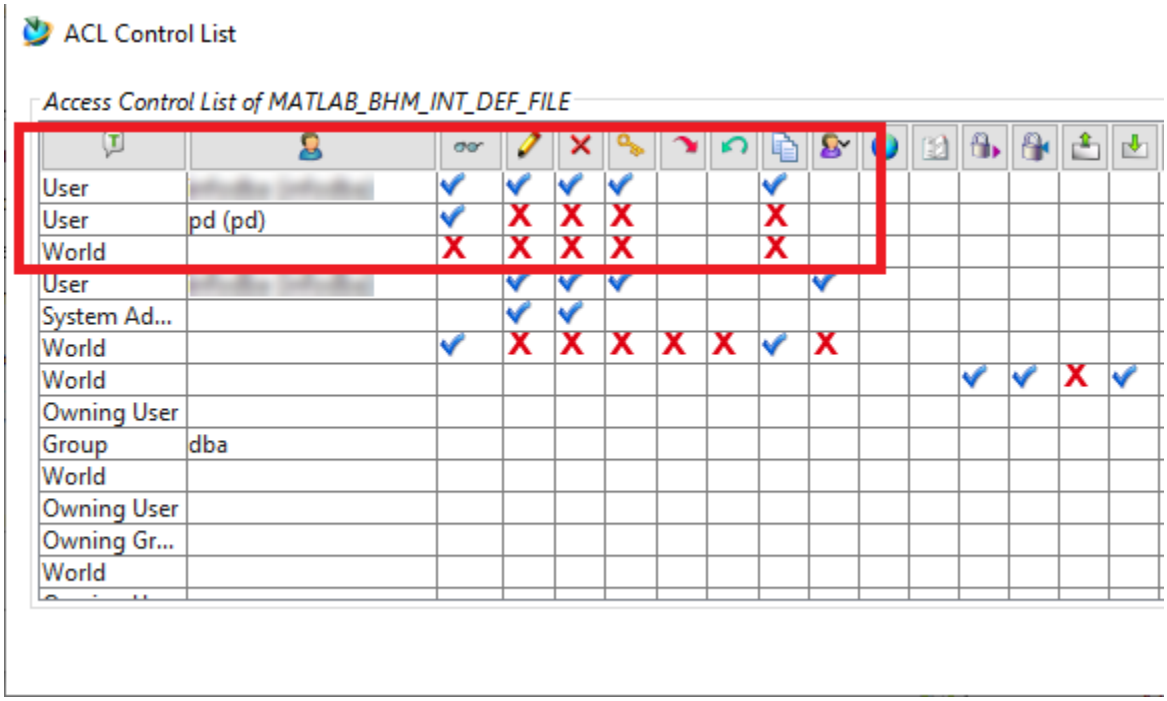
3. Check in the dataset to Teamcenter.

4. Log on to the System Modeling Workbench integration client to use the updated configuration.

Configure integration definition files for different users

You can change the behavior of integration operations for different group of users by creating different integration definition files for each group. To do this, create different integration definition files with the same name and then set access rules to different groups as per your requirement. When giving access permissions, remember the following:

- Block all permissions for the world
- Enable all permissions for Teamcenter admin user.
- Give read permission to the group



After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Set up common connector-based integration

To set up the common connector-based integration, perform the following two configurations:

- Update the project definition file with information about the modeling tool file types
- Create an integration definition file that contains the mapping information between the modeling tool types and Teamcenter types. If an integration already exists, for example, the MATLAB Simulink integration, you need not create an integration definition file. You can modify the existing one.

Update the project definition file with information about the modeling tool file types

1. In the Teamcenter rich client, search for and check out the dataset **PROJECT_BHM_INT_DEF_FILE**.
2. Update the named reference file in the dataset by adding information about the modeling tool and its file extension in the **ToolFileMapping** section as follows:

```
<ToolFileMapping>
  <ToolFileMap toolType="MATLAB" fileExtension="mdl"/>
  <ToolFileMap toolType="MATLAB" fileExtension="slx"/>
  <ToolFileMap toolType="<CUSTOMTOOLTYPE>" fileExtension=
  "<Extsion of file specific to tool>"/>
</ToolFileMapping>
```

3. Check in the **PROJECT_BHM_INT_DEF_FILE** dataset.

Create an integration definition file that contains mapping information between the modeling tool types and Teamcenter types

1. Create an XML file named *CUSTOMTOOLTYPE_BHMIntegrationDefinition.xml*.
2. Update the XML file with mapping information.
3. If you are modifying an existing integration definition file, update the **name** attribute of the **ConnectorClass**.
4. Create a dataset in Teamcenter named *CUSTOMTOOLTYPE* **_BHM_INT_DEF_FILE** and add the XML file that you created to the dataset as a named reference.
5. Use the command line operation to import and export the model data.

The *CUSTOMTOOLTYPE_BHMIntegrationDefinition.xml* definition is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<BHMIntegration xmlns="http://www.plmxml.org/Schemas/bhm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.plmxml.org/Schemas/bhm"
applicationName="%CUSTOMETOOLTYPE%">
  <!--Below line specifies to use smart connector-->
  <ConnectorClass fullName="com.teamcenter.behaviormodeling.commonclient.
  defaultconnector.CommonConnector" jarFilePath="" />
  <SupportedVersions>
    <Version id="17" />
  </SupportedVersions>
  <!-- Map object type and teamcenter type with behavior-->
  <ObjectMapping type="Model" behaviorType="BVR/GRM"
tctype="%teamcentertype%">
    <BHMElement type="RootModel" tctype="%teamcentertype%">
      <AttributeMappings>
        <AttributeMapping name="Name" tcattr="object_name"
           includeinduplicatecheck="false"/>
        <RevisionAttributeMapping/>
      </AttributeMappings>
    </BHMElement>
    <BHMElement type="Project" checkforduplicates="false"
reftype="Model">
      <AttributeMappings>
        <AttributeMapping name="Name" tcattr="object_name"
              includeinduplicatecheck="false"/>
        <RevisionAttributeMapping/>
      </AttributeMappings>
    </BHMElement>
    <OrganizationData>
    </OrganizationData>
  </ObjectMapping>
  <PrimaryDataFileMapping>
    <FileMapping>
    <!-- Map file extension of tool type with supported dataset and
named
    reference-->
      <FileMap fileExt="file extension same as what was mapped in the
      PROJECT_BHM_INT_DEF_FILE  ToolFileMapping section"
      tcDatasetType=
      "supported teamcenter dataset for file extension"
      tcNameReferencedType=
      "supported teamcenter dataset names reference for file
extension" />
      <FileMap fileExt="*" tcDatasetType=
      "supported teamcenter dataset for generic file e.g MISC"
      tcNameReferencedType=
      "supported teamcenter dataset names reference for file extension
      e.g. MISC_TEXT" />
    </FileMapping>
```

```
  </PrimaryDataFileMapping>
  <FileMapping><!--additional mapping for File-->
    <FileMap fileExt="txt" tcDatasetType="Text"
tcNameReferencedType="Text" />
    <FileMap fileExt="png" tcDatasetType="Image"
tcNameReferencedType="Image" />
    <FileMap fileExt="html" tcDatasetType="HTML"
tcNameReferencedType="HTML" />
    <FileMap fileExt="jpg" tcDatasetType="JPEG"
tcNameReferencedType="JPEG_Reference" />
    <FileMap fileExt="zip" tcDatasetType="Zip"
tcNameReferencedType="ZIPFILE" />
    <FileMap fileExt="tif" tcDatasetType="TIF"
tcNameReferencedType="TIF_Reference" />
    <FileMap fileExt="gif" tcDatasetType="GIF"
tcNameReferencedType="GIF_Reference" />
    <FileMap fileExt="*" tcDatasetType="MISC"
tcNameReferencedType="MISC_TEXT" />
  </FileMapping>
  <Extensions>
  </Extensions>
  <Operations>
  </Operations>
</BHMIntegration>
```

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable the Active Workspace Open in Tool command

To enable the opening of modeling tools from Active Workspace, install the following feature in Teamcenter Environment Manager (TEM):

Base Install→Active Workspace→Client→MBSE services Active Workspace Client.

If using Deployment Center, **Open in Tool** is automatically installed when you install either any MBSE application or integration.

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.

- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Configure a verification request

1. **Create the analysis definition**

   Each verification request is based on a template called the *analysis definition*. You must create an analysis definition for each request type that a user can create in your business.

   You specify each analysis definition in a separate XML file that lists the input and output item types permitted in an verification request.

   A default analysis configuration file is created whenever an analysis definition object is created. You can replace or remove the configuration file if necessary. However, you must validate a new file against the specified XML schema.

   An example of the required format is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
-<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Crt0ContractPkgSchema.xsd">
  -<input>
    <objectType type="Fnd0LogicalBlockRevision" quantity="*"/>
    <objectType type="Requirement Revision" quantity="*"/>
    <objectType type="RequirementSpec Revision" quantity="*"/>
    <objectType type="DocumentRevision" quantity="*"/>
    <objectType type="ItemRevision" quantity="*"/>
  </input>
  -<output>
    <objectType type="DocumentRevision" quantity="*"/>
    <objectType type="ItemRevision" quantity="*"/>
  </output>
</configuration>
```

2. **Update the SOA mapping file**

   The **Get Analysis Request** API does not return the properties for certain parameters, such as properties for **object_name**, **object_desc**, and **object_type** in the Verification Request Revision object. To resolve this, add the required objects and their properties to the *ARObjectPropertyConfiguration.xml* file. This file is located in the *TC_ROOT/bhm/config* folder.

   > Example:
   >
   > For adding the Verification Request Revision object and its properties, add the following lines to the file:

```
<Object Type="Crt0VldnContractRevision" >
    <Property Name="object_name" />
    <Property Name="object_type" />
    <Property Name="object_desc" />
    <Property Name="crt0Configuration" />
    <Property Name="crt0Domain" />
    <Property Name="crt0Purpose" />
    <Property Name="crt0Result" />
    <Property Name="crt0State" />
</Object>
```

3. **Update the BHM client policy file**

   If you want additional properties of an object returned by the **Get Analysis Request** API, modify the *BHMClient.policy* file on the server as follows:

   a. Navigate to the *TC_ROOT/tcdata/soa/policies* folder.

   b. Open the *BHMClientPolicy.xml* file.

   c. Update the following in the **ObjectType** section of **Awb0Connection**:

```
<ObjectType name="Awb0Connection" >
    <Property name="awb0End1" withProperties="true" />
    <Property name="awb0End2" withProperties="true" />
</ObjectType>
```

   d. Make similar changes to the *BHMClientPolicy.xml* file located in the *TC_DATA/soa/policies* folder.

4. **Client side verification request mapping**

   Configure the files and datasets that can be associated with the verification request results by updating the *AR_IntegrationDefinition.xml* file.

   This file is located in the *TC_ROOT\bhm\config* folder.

5. **Update the verification request style sheet**

   Update the verification request style sheet to display the datasets in the **Output** section of the **Results** tab.

   The style sheet is named *Crt0VldnContractRevSummary.xml*. Search for the dataset *Crt0VldnContractRevSummary* and modify the associated named reference XML file.

Update the **tc_xrt_Reports** section with the dataset types.

Example:

To add a PDF type, update the **source** attribute of **objectSet** as a comma-separated entry:

```
Crt1AnalysisRequestOutProvider.Text,
Crt1AnalysisRequestOutProvider.PDF
```

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Classify models

- Create a library for reusing models, using the Classification Admin application.

  Classify the models in the library, using the Classification application.

  For information about creating libraries, see *Basic Classification — Deployment and Administration* in the Teamcenter help.

  For information about classifying models, see *Basic Classification on Rich Client — Usage* in the Teamcenter help.

- To configure which classification libraries appear in Teamcenter integration dialog boxes of the modeling tool application, update the **Bhm0FilterEntriesForBehaviorModl** preference.

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable the sample classification library for model management

A sample classification library called Model Identity Card is available for model management. To get this library, do the following:

1. Update the value of the **CLS_is_presentation_hierarchy_active** preference to true.

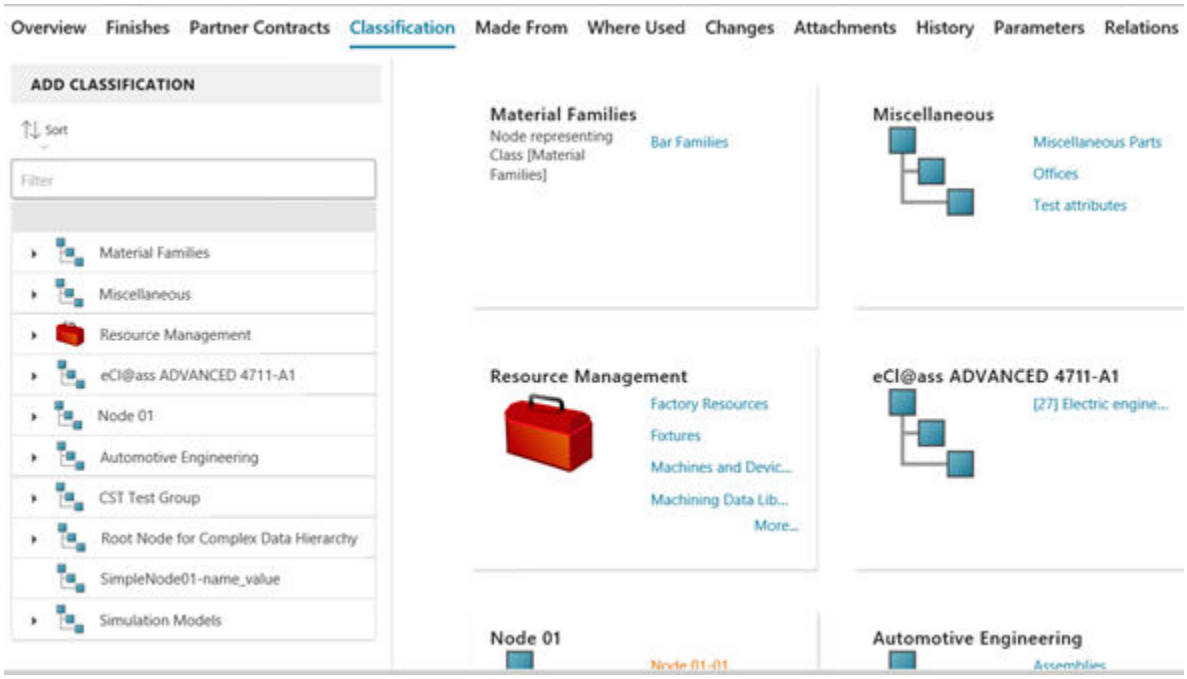
2. In the MBSE Integration Gateway kit, go to the **additional_applications** folder and extract the *modelidentitycard* ZIP file to a location of your choice.

3. Execute the following commands using the Teamcenter command prompt. Ensure that you update the path *D:\share* to the path where you extracted the *modelidentitycard* ZIP file:

```
clsutility -u=Tc-admin-user -p=password -g=group -create
-keylov_definitions -request="D:\share\MIC\TcME_MIC\MICLOV.json"

clsutility -u=Tc-admin-user -p=password
-g=group -create -property_definitions
-request="D:\share\MIC\TcME_MIC\MICProperty.json"

clsutility -u=Tc-admin-user -p=password -g=group -create
-class_definitions -request="D:\share\MIC\TcME_MIC\MICBlocks.json"

clsutility -u=Tc-admin-user -p=password -g=group -create
-class_definitions -request="D:\share\MIC\TcME_MIC\MICClass.json"

clsutility -u=Tc-admin-user -p=password -g=group -create
-node_definitions -request="D:\share\MIC\TcME_MIC\MICNodes.json"

clsutility -u=Tc-admin-user -p=password -g=group -update -status
-request="D:\share\MIC\TcME_MIC\ReleaseStatus.json"

bmide_modeltool.bat -u=Tc-admin-user -p=password -g=group
-tool=all -mode=upgrade -target_dir="%TC_DATA%"
%TC_ROOT%\solr-7.7.0\TcSchemaToSolrSchemaTransform.bat
%TC_DATA%\ftsi\solr_schema_files  %TC_ROOT%/TcFTSIndexer/bin/
runTcFtsIndexer.bat -task=objdata:clear 4 %TC_ROOT%/TcFTSIndexer/bin/
runTcFtsIndexer.bat -task=objdata:index  %TC_ROOT%/TcFTSIndexer/bin/
runTcFTSIndexer -task=objdata:sync -interval=30
```

4. After running the utility, type **4** in the command prompt when the prompt `0-7` comes up.

Once you run these commands, the sample classification library, Model Identity Card, is imported. You can view this in the **Classification** tab of Active Workspace.



After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Configure the data to be excluded during import

You can configure the import operation to exclude certain files and folders. This exclusion is done using configuration files.

You can specify the configuration at the administrator level or at the user level by modifying the following files:

- To specify exclusion at the administrator level, update the *PROJECT_BHMintegrationDefinition.xml* file that is associated with the *PROJECT_BHM_INT_DEF_FILE* dataset as a named reference.
- To specify exclusion at the user level, update the *User_Specific_BHMIntegrationDefinition.xml* file located in the *TCME_install/bhm/config* directory on the user's computer.

Update the **Exclusions** node in the configuration file with the relative path of the exclusions.

Example:

```
<Exclusions>
  <RelativePath>*.txt</RelativePath>
  <RelativePath>/Model/*.slx</RelativePath>
  <RelativePath>/Model/test</RelativePath>
</Exclusions>
```

You can specify the following exclusions:

- Explicit list of folders

  Example: `<RelativePath>/Model/test</RelativePath>`

- Explicit list of files

  Example: `<RelativePath>MRS_Exc_Model0_g.slx</RelativePath>`

- Explicit list of file extensions

  Example: `<RelativePath>*.txt</RelativePath>`

For Git integration, an additional folder is added, which makes the root folder different from any other integration folder. Hence when using Git integration, ensure that you specify the correct relative path.

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Create behavior modeling objects in Teamcenter

In addition to the behavior modeling objects available in Teamcenter, you can create custom objects (with custom attributes) that correspond to the behavior modeling tool objects. You can create custom objects using Business Modeler IDE.

The **Bhm0BehaviorModl** class must be the parent of the new class you create in Teamcenter.

For more information, see *BMIDE for Data Model Design* in the Teamcenter help.

After you create the custom objects, you must define these objects in the integration definition file.

After completing this configuration, you can:

- Refer to the **List of optional server-side configurations for Teamcenter MBSE Integration Gateway** topic to see other configurations.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Configure how long object names and occurrence names are handled

The **TCME_TRUNCATE_LONG_NAMES** preference controls how long object names and occurrence names are handled. The default value of this preference is **FALSE**.

If the value of this preference is **FALSE**, an error occurs, and the operation ends when object names are greater than 128 characters or occurrence names are greater than 100 characters.

If the value of this preference is **TRUE**, object names that are greater than 128 characters are truncated to 128 characters. Occurrence names greater than 100 characters are truncated to 100 characters.

Step 5: Install Teamcenter MBSE Integration Gateway client on a desktop

Prerequisites to installing the Teamcenter MBSE Integration Gateway desktop client

Ensure that you complete the following tasks before installing the Teamcenter MBSE Integration Gateway desktop client:

- Install a supported version of JRE and JDK.
- Set the following environment variables:
  - Set the value of the **JAVA_HOME** variable to the JDK home folder path.
  - Set the value of the **JRE_HOME** variable to the JRE home folder path.
  - Update the value of the **PATH** variable with the following value:

    ```
    %JRE64_HOME%\bin;%JAVA_HOME%\bin;%PATH%
    ```
- Install the Teamcenter server. Ensure that the Teamcenter server URL and the FMS URL are available.
- Ensure that the user performing the installation has administrative rights on the machine.
- Ensure that the following installation kits are available:

  - Teamcenter Foundation kit
  - (Optional) Teamcenter Foundation patch kit — this depends on which version is to be installed

    The downloaded kits must match the version of the Teamcenter server. For example, if the Teamcenter server version is 10.1.4, download the Teamcenter 10.1 and 10.1.4 kits.

- Ensure that the folder specifying the staging directory is already created.
- For Simulink and SMW integrations, ensure that the respective (compatible) desktop versions are installed.
- For Security Services, ensure that the **Login URL** and the **SSO App ID** values are available.
- Ensure that security certificates are installed in the JVM securities directory for HTTPS setup.

After performing this task, you can **install the Teamcenter MBSE Integration Gateway desktop client.**

Install the Teamcenter MBSE Integration Gateway desktop client using TEM

This topic covers the installation of the following features:

- Simulink Integration
- System Modeling Integration (SMW-Capella)
- Software Management

1. As an administrator, launch Teamcenter Environment Manager from the Teamcenter foundation kit.
2. In the **Install/Upgrade Options** panel, click **Install**.
3. In the **Media Locations** panel, specify the location of the Teamcenter install file in the **Update Location** list.
4. Click **Next** until you reach the **Features** panel.
5. In the **Features** panel, expand **Extensions→MBSE Integrations**, expand **Client**, and select:
   - **Simulink Integration** to install the MATLAB integration connector.

     Note:

     This feature is deprecated and will be obsoleted in a future release.

- **System Modeling Integration** to install system modeling integration support.

  To install this option, you must also choose the **Gateway for modeling** option.

- **Teamcenter Behavior Modeling for Rich Client** to install behavior modeling support on the rich client.

- **Teamcenter MATLAB RAC** to install MATLAB integration support on the rich client.

  To install this option, you must also choose the **Simulink Integration** option.

  > Note:
  >
  > This feature is deprecated and will be obsoleted in a future release.

- **Software Management** to install software management support.

  To install this option, you must first choose the **MBSE service Client** option.

Specify the location to install the client in the **Installation Directory** box. This becomes the root location.

6. In the **File Client Cache (FCC)** panel, select **Use New FCC** when you are installing a new client.

7. In the **FCC Parents** panel, specify the details of the Teamcenter server and click **Next**.

8. In the **MATLAB Client Information** panel, do the following:

   a. In the **MATLAB Installation Directory** box, type or browse to the location where MATLAB is installed. This must be the directory containing the **bin** directory. This becomes the MATLAB root directory.

   b. In the **Staging Directory** box, type or browse to the location where models downloaded from Teamcenter are stored, for example, **C:\StagingDir\MATLAB**

9. In the **MBSE Integration Gateway Common Client** panel, type the Teamcenter server URL and click **Next**.

10. In the **Confirmation** panel, click **Start**.

11. When the update is complete, click **Close**.

After performing this step, proceed to the **post-installation configuration tasks for the Teamcenter MBSE Integration Gateway desktop client.**

Install the Teamcenter MBSE Integration Gateway desktop client using Deployment Center

Add the Teamcenter MBSE Integration Gateway desktop client to your existing Teamcenter environment.

1. Log on to Deployment Center and select the environment to which you want to add MBSE Integration Gateway client.

2. Go to the **Applications** task. Click **Add or Remove Selected Applications** ✎.

3. In the **Available Applications** panel, use the web browser search to find the following applications:

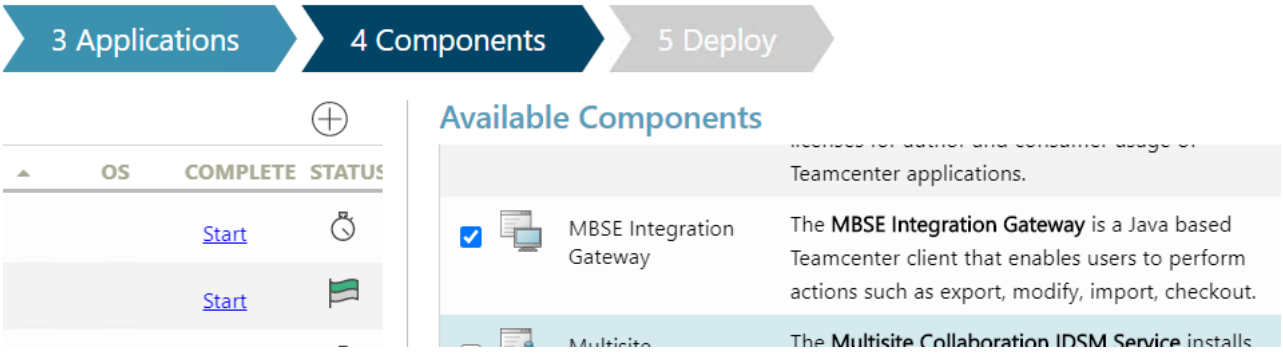
Integration	Application to choose
Software Management	**Software Management**
System Modeling Integration	**System Modeling Workbench Integration**
Enterprise Model Management This installs the generic integration that allows you to import and work with models stored in your file system into Teamcenter. This also installs the integration for Git.	**Support for Concurrent Modeling**

   Select the application, and then click **Update Selected Applications**.

   Deployment Center automatically selects any additional dependent applications.

4. Go to the **Components** task.

5. In the **Components** task, click **Add component** ⊕ and select **MBSE Integration Gateway**.



   Click **Update Selected Components**.

6. Update the **MBSE Integration Gateway** component as follows:

Entry	Description
Enable Mass Client Deploy	Select this option if you want to deploy the MBSE Integration Gateway client on multiple machines.
Machine Name	• In the **Machine Name** box, specify the machine where the MBSE Integration Gateway client will be installed. • In the **OS** list, select the operating system.
General Settings	In the **Teamcenter MBSE Integration Gateway Installation Path** box, specify the path where the MBSE Integration Gateway client will be installed.
MBSE Integration Gateway Common Client	In the **URL** box, specify the URL of the Teamcenter server.

7. Update the **Teamcenter Client Communication System** component.

8. In the **Selected Components** list, note any remaining components whose configuration status is not **100%**. Select each incomplete component, enter required parameters, and save component settings until all components in the environment show a configuration status of **100%**.

   When all components are fully configured, the **Deploy** task is enabled.

9. Once you have completed all the tasks, you can deploy the environment.

After performing this step, proceed to the **post-installation configuration tasks for the Teamcenter MBSE Integration Gateway desktop client.**

Post-installation configuration tasks for the Teamcenter MBSE Integration Gateway desktop client

Ensure that the following configurations are done after installing the Teamcenter MBSE Integration Gateway desktop client:

- If accessing a Teamcenter server hosted on a virtual machine cloud, add the server IP and the domain name in the *hosts* file located at *C:/Systems32/Drivers/etc* or *C:\Windows\System32\drivers\etc*.

  Example: `10.134.56.84 vc6s004.net.plm.eds.com`

- In the *BHMClient.properties* file located in the *TC_MBSE_Client_ROOT* **\bhm** directory, set the value of **PROJECT.stagingDir** to the same local directory as for the other tool integrations.

```
#==============================================================================
# BHM File System Variables
# -------------------------
```

```
# Set the following variables to define the cache and staging
directories on the
# operating system.
# Path separator for unix should be "/", and "\\" for windows.
# MATLAB.StagingDir - Defines the staging directory to be used by BHM.
Required.
#==========================================================================

MATLAB.StagingDir=C:\\bhm\\staging
PROJECT.StagingDir=C:\\bhm\\staging
```

- For hosted Active Workspace, update the **ActiveWorkspaceHosting.URL** preference with the value of the Active Workspace url.

  Example:

  http://*hostname*:3000

  The value http or https should be based on the server.

- For Security Services, ensure that the following variables are defined in the *CommonClient.properties* file:

```
#==========================================================================
# SSO Connection Variables
# ------------------------
# Set the following variables to define the target Teamcenter SSO
server,
# Application ID and SSO Session flag value.
# Note : These are the default values and will be changed as per the
# SSO login dialog.
#
# TC_SSO_LOGIN_URL    - Defines the default Teamcenter SSO URL for the
login dialog.
# TC_SSO_APP_ID       - Defines the default Teamcenter SSO Application
ID.
# TC_SSO_SESSION_FLAG - Defines the default Teamcenter SSO Session
Flag.
#
#==========================================================================

TC_SSO_LOGIN_URL=http://<servername>:<port>/sa
TC_SSO_APP_ID=<SSO-App-ID>
TC_SSO_SESSION_FLAG=true
```

> Note:
>
> Installation of the security agent is required on the desktop client for Security Services to work.

- Check if fcc for the MBSE Integration Gateway client is started using the command:

  ```
  fccstat.exe -status
  ```

  If it is not started, open the folder *TC_MBSE_Client_ROOT* **\tccs\bin** and start the *fccstat* command from the command line as follows:

  ```
  fccstat.exe -start
  ```

- Other points to note:

  - One MBSE Integration Gateway client can point to only one Teamcenter server. To connect to multiple Teamcenter servers, multiple MBSE Integration Gateway client installations are required.
  - The MBSE Integration Gateway client version must match the MBSE Integration Gateway server version.

For additional configurations to the MBSE Integration Gateway client, see **Step 6: Performing optional client-side configurations for Teamcenter MBSE Integration Gateway**.

After performing these configurations, for the next steps, refer to the **MBSE Integration Gateway deployment checklist**.

Step 6: Performing optional client-side configurations for Teamcenter MBSE Integration Gateway

List of optional client configurations for Teamcenter MBSE Integration Gateway

You can perform the following optional configurations for the Teamcenter MBSE Integration Gateway client:

Configuration task	Purpose
Set up context and target folders	To specify the location of the context folder or parent folder and the target folder or the folder where the models are saved
Set up the cache directory	To specify the location of the derby cache directory
Enable data logging	To specify where to store the data logs

Configuration task	Purpose
Enable support for multiple instances of Teamcenter MBSE Integration Gateway	To enable support for multiple instances of Teamcenter MBSE Integration Gateway clients
Enable the Active Workspace Open in Tool command	To enable the Open in Tool command on Active Workspace
Update the *mmgenv.bat* file with Teamcenter variable information	To specify Teamcenter variables required for integration

After performing these configurations, for the next steps, refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Set up context and target folders

In the *CommonClient.properties* file, specify the context and target folders as follows:

- Context folder: Update the value of the **DefaultContextFolder** with the context folder location.

  Example:

  ```
  DefaultContextFolder=D:\\Models\\MyModelRoot
  ```

- Target folder: Update the value of the **DefaultTargetFolder** with the target folder location.

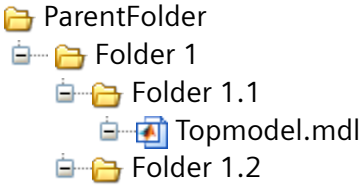
  Example:

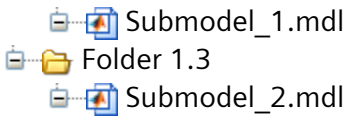
  ```
  DefaultTargetFolder=D:\\Staging\\DefaultTargetFolder
  ```

Default context folder

The default context folder allows you to specify a common parent folder under which all models and model folders are organized. The context folder is used during the initial save, import, and add operations.

As an example, consider the following folder hierarchy:





Suppose that the context folder is set as *D:\ParentFolder* and you save the contents of *Folder 1.1* to Teamcenter. Teamcenter picks up the context folder information and organizes the saved model file as follows:

```
ParentFolder-->Folder 1-->Folder 1.1-->Topmodel.mdl
```

If you do not specify a context folder, the files are organized based on the parent folder.

Default target folder

The default target folder defines the folder hierarchy of the models in the staging directory. The target folder is used during the save as, new save, and revise operations.

As an example, consider that your staging directory is *C:/Staging* and you intend to download the models inside the directory *Parent Level/MidLevel/Car/PT/Engines*. When you open a model, the model is downloaded to the staging-directory at *C:/Staging/Parent Level/MidLevel/Car/PT/Engines*.

You can also specify the target folder when performing the operations mentioned above. If you do not specify the target folder, the existing behavior continues, that is, the models are saved in the *Staging-directory/model-name_itemid_revisionid* folder.

How is the folder where models are downloaded calculated based on context and target folders?

The location where your model files are downloaded is calculated based on the values of the default context folder and default target folder. The path where the models are downloaded is the concatenation of the target folder and the context folder.

Result folder=Target-folder+Context-folder

> Example:
>
> Assume that your Simulink models are located in the directory structure *D:\models\ContextExample\Level1\Level2\Level3\Level4* and that your staging directory is located at *D:\staging* . The following table shows the path to the folder where the models are downloaded based on how you set up the context and target folders:

Context folder	Target folder	Result folder
NA	NA	*D:\staging\ ModelName_ItemID_RevID*
D:\models\-ContextExample\Level1	NA	*D:\staging\ModelName_ItemID_RevID\Level1\Level2\Level3\Level4*
NA	*D:\staging\-MyWorkspace*	*D:\staging\MyWorkspace*
D:\models\-ContextExample\Level1	*D:\staging\-MyWorkspace*	*D:\staging\MyWorkspace\Level1\Level2\-Level3\Level4*

After completing this configuration, you can:

- Check if you need to perform any other **client configurations for Teamcenter MBSE Integration Gateway**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Set up the cache directory and local caching

Setup the cache directory

The default Derby cache location is defined in the *BHMClient.properties* file. This file is located in the *TC_ROOT\bhm* directory. Check the file to ensure that the cache directory is defined. If not, define the cache directory by updating the following entry:

- **CacheDir=***folder-location-on-the-client-machine**derby-cache-folder-name*.

After completing this configuration, you can:

Choose if local caching should be disabled

You can choose if you want to disable the data cache locally on your client. By default, files are locally cached.

If you want disable the local cache, update the *BHMClient.properties* file located in *TC_ROOT/bhm* folder and add the following entry:

```
TC_Client_Cache=false
```

To enable local cache, update the entry in the *BHMClient.properties* file to:

```
TC_Client_Cache=true
```

By default, the files are cached in *TC_ROOT/bhm/DataCacheDir*. You can change the data cache directory by adding the following entry to the BHMClient.properties file:

`Cache_Dir=`*location-of-the-cache-dir*

- Check if you need to perform any other **client configurations for Teamcenter MBSE Integration Gateway**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable data logging

Teamcenter MBSE Integration Gateway uses the log4j mechanism for data logging. You can change the logging parameters in the *log4j.properties* file specified in the *TC_ROOT/bhm* directory.

By default, the log file is saved in the *system temp dir/TCMEGateway.log* directory. You can change this location by modifying the *log4j.properties* file located in the *TC_ROOT/bhm* folder.

For more information about log4j, see the Apache log4j documentation.

After completing this configuration, you can:

- Check if you need to perform any other **client configurations for Teamcenter MBSE Integration Gateway**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable support for multiple instances of Teamcenter MBSE Integration Gateway

To enable support for multiple instances of Teamcenter MBSE Integration Gateway, add an entry in the client cache section for each modeling tool in the *BHMClient.properties* file. This file is located in the *TC_ROOT\bhm* directory. Update the file in one of the following formats:

- *ToolName*.**CacheDir="***folder-location-on-the-client-machine***"**.

Example:

```
MATLAB.CacheDir="D:\Apps\MatlabCacheDir"
```

OR

ToolName.**CacheDir="%***environment-variable***%***folder-location-on-the-client-machine***"**.

Example:

```
MATLAB.CacheDir="%TC_ROOT%\D:\Apps\MatlabCacheDir"
```

After completing this configuration, you can:

- Check if you need to perform any other **client configurations for Teamcenter MBSE Integration Gateway**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Enable the Active Workspace Open in Tool command

To enable the opening of modeling tools from Active Workspace, install the following feature in Teamcenter Environment Manager (TEM):

Extensions→Model Management→Active Workspace→Open in Tool.

If using Deployment Center, install the following feature:

Teamcenter→Foundation→MBSE Integration Gateway→Model Management Active Workspace→Open in Tool.

After completing this configuration, you can:

- Check if you need to perform any other **client configurations for Teamcenter MBSE Integration Gateway**.
- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Update the mmgenv.bat file with Teamcenter variable information

After installing the behavior modeling client on your machine, you must ensure that the *mmgenv.bat* file is updated with information about Teamcenter variables.

The *mmgenv.bat* file is located in the *TC_ROOT\bhm* directory.

Ensure that the following variables are set:

```
set TC_ROOT=location-of-TC_ROOT
set MATLAB_ROOT=location-of-MATLAB_ROOT
set FMS_HOME=location-of-FMS_HOME
```

Example:

```
set TC_ROOT=D:\apps\Siemens\TcMEPostEAP
set MATLAB_ROOT=D:\apps\MATLAB\R2016a
set FMS_HOME=D:\Apps\Siemens\TcMEPostEAP\tccs
```

After completing this configuration, you can:

- Refer to the **MBSE Integration Gateway deployment checklist** to see next steps.

Step 7: Test the integrations

Verify Enterprise Model Management

After deploying Enterprise Model Management, you can:

- Import Git data into Teamcenter
- Update Git data in Teamcenter
- Download Git data from Teamcenter

For more information on performing these tests, see *Enterprise Model Management in Active Workspace documentation*.

6. Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features

Step 1: Perform the prerequisite steps

- Based on the modeling tool that is currently integrated with, refer to the Hardware and Software Certifications knowledge base article on https://support.sw.siemens.com to check the version of Teamcenter you need to update or upgrade to.
- Update or upgrade to the version of Teamcenter that supports the version of the modeling tool you want integrated with Teamcenter.

After completing this configuration, you can:

- Perform **Step 2: Update the Teamcenter MBSE Integration Gateway framework and server components**.
- Refer to the checklist **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** to see next steps.

Step 2: Update the Teamcenter MBSE Integration Gateway framework and server components using TEM

If you are updating from Teamcenter MBSE Integration Gateway version 4.1 to a higher version, you must **perform certain configurations**.

When you update Teamcenter, Teamcenter MBSE Integration Gateway is also updated.

If you need to install new Teamcenter MBSE Integration Gateway features, see *Step 2: Install Teamcenter MBSE Integration Gateway framework and server features using TEM*.

After you have updated the server features, you can check if the integration features on the Teamcenter server are configured. Choose from the following configuration topics:

- **Check if you have configured Enterprise Model Management**
- **Check if you need to perform optional server configurations for Teamcenter MBSE Integration Gateway**

You can also refer to the checklist **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features**.

Step 2: Update the Teamcenter MBSE Integration Gateway framework and server components using Deployment Center

This topic documents how you can update the Teamcenter MBSE Integration Gateway framework and server components using Deployment Center.

Procedure

1. When you update Teamcenter, Teamcenter MBSE Integration Gateway is also updated.

   If you need to install new Teamcenter MBSE Integration Gateway features, see *Step 2: Install Teamcenter MBSE Integration Gateway framework and server features using Deployment Center*.

Postrequisites

After you have updated the server features, you can check if the integration features on the Teamcenter server are configured. Choose from the following configuration topics:

- **Check if you have configured Enterprise Model Management**
- **Check if you need to perform optional server configurations for Teamcenter MBSE Integration Gateway**

You can also refer to the checklist **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features**.

Step 2.1: Configurations to be done before updating from Teamcenter MBSE Integration Gateway version 4.1 to a higher version

Before upgrading Teamcenter MBSE Integration Gateway version 4.1 to a higher version, perform the following steps:

1. Create a file named *change_dataset.xml*.

2. In the XML file you created, ensure the following content exists:

```
<TcDatasets>
  <TcDataset name="Sys0CapellaModel">
    <TcTool fromName="Sy0SysMLTool" toName="Sys0CapellaTool"/>
    <TcDatasetReferences>
      <TcDatasetReference fromName="Sym0SYSMLZIP"
toName="Sys0CAPZIP"/>
```

```
      <TcDatasetReference fromName="Sym0SYSMLMISC"
toName="Sys0CAPMISC"/>
    </TcDatasetReferences>
  </TcDataset>
</TcDatasets>
```

3. Ensure that there are no active Teamcenter server connections.

4. Run the following commands:

   a. `%TC_BIN%\change_type_name -u=`*Tc-admin-user* `-p=`*password* `-g=`*group* `-old_type=Sym0SysMLModel -new_type=Sys0CapellaModel`

   b. `%TC_BIN%\change_datasets -u=`*Tc-admin-user* `-p=`*password* `-g=`*group* `-ds_info=change_dataset.xml`

   c. `%TC_BIN%\install -regen_schema_file -u=`*Tc-admin-user* `-p=`*password* `-g=`*group*

   d. `%TC_BIN%\generate_metadata_cache -u=`*Tc-admin-user* `-p=`*password* `–force`

After performing the previous steps, you can **update the Teamcenter MBSE Integration Gateway framework and server components**.

Step 3: Check if you have performed the mandatory server configurations for your desired integration

Check if you have performed Enterprise Model Management configurations

You can perform the following configurations for Git and file system integrations:

- **Enable the Git or file system integrations**
- **Enable display of image preview for models and model collections**
- **Configure additional properties for import**
- **Remove Git credentials when uninstalling MBSE Integration Gateway**

After you are done with performing these configurations, refer to the **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** to see next steps.

Step 4: Check if you have perfomed the optional server-side configurations for Teamcenter MBSE Integration Gateway

Check if you have performed the following optional server-side configurations for Teamcenter MBSE Integration Gateway:

Configuration	Description
Mapping data using integration definition file	To customize how the data is mapped between Teamcenter and the modeling tool
Configure integration definition file for different users	To customize the integration behavior for different users or groups
Setup common connector-based integration	To enable integration with the tool of your choice when you are using the generic or common connector-based integration
Enable the Active Workspace Open in Tool command	To enable the Open in Tool command in Active Workspace that allows you to open models in Active Workspace the respective modeling tools
Configure a verification request	To enable the use of Verification Request with MBSE Integration Gateway
Classify models	To setup the classification of models in Teamcenter
Enable the sample classification library for model management	To enable the sample classification library to manage your models in Teamcenter
Configure the data to be excluded during import	To choose what data is excluded during import
Create behavior modeling objects in Teamcenter	To create custom behavior modeling objects in Teamcenter

After you are done with performing these configurations, refer to the **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** to see next steps.

Step 5: Update Teamcenter MBSE Integration Gateway desktop client using TEM

Before you update MBSE Integration Gateway desktop client, ensure you do the following:

- Check the Hardware and Software knowledge base article on https://support.sw.siemens.com to identify which version of Teamcenter the new MBSE Integration Gateway desktop client supports. Upgrade to the supported version of Teamcenter.

- Update to the supported version of Teamcenter.
- If upgrading from MBSE Integration Gateway version 4.3 to MBSE Integration Gateway version 5.0, delete the derby cache located in the *TC_ROOT\bhm\dataCacheDir\db\bhmCache* directory.

Updating MBSE Integration Gateway client

1. From Support Center, download and extract the contents of the Teamcenter patch zip file to any folder.
2. Launch Teamcenter Environment Manager from the *MBSE Integration Gateway-client-root/install* directory as an administrator.

   This *MBSE Integration Gateway-client-root/install* directory is the place where you installed the MBSE Integration Gateway client.
3. In the **Apply Updates** panel:
   a. In the **Update kit location** box, type or browse to the location where you extracted the contents of Teamcenter ZIP file.
   b. In the **Backup directory** box, type or browse to the location where you wish to create the backup files.
   c. In the **Original Media** panel, type or browse to the location of the base Teamcenter install kit.
   d. Click **Next**.
4. Click **OK** to close the **Patch information** window.
5. Click **Close** to close the **Status Message** window.
6. In the **Confirmation** panel, click **Start**.
7. When the update is complete, click **Close**.

Once the update is complete, you can install new MBSE Integration Gateway features as follows:

Installing MBSE Integration Gateway features

1. Launch Teamcenter Environment Manager from the *MBSE Integration Gateway-client-root/install* directory as an administrator.

   This *MBSE Integration Gateway-client-root/install* directory is the place where you installed the MBSE Integration Gateway client.

2. In the **Maintenance** panel, select **Configuration Manager** and click **Next**.

3. In the **Configuration Maintenance** panel, select **Perform maintenance on an existing configuration** and click **Next**.

4. In the **Old Configuration** panel, select a configuration and click **Next**.

5. In the **Feature Maintenance** panel, select **Add/Remove Features** and click **Next**.

6. In the **Features** panel, select the MBSE Integration Gateway client features from **Extensions→Model Management →Client**.

   After selecting all the client features, click **Next**.

7. In the **Confirmation** panel, click **Start** to install the client features.

Refer to the checklist **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** for next steps.

Step 5: Update the Teamcenter MBSE Integration Gateway desktop client using Deployment Center

This topic documents how you can update the Teamcenter MBSE Integration Gateway desktop client using Deployment Center.

Prerequisites

Before you update MBSE Integration Gateway desktop client, ensure you do the following:

- Check the Hardware and Software knowledge base article on https://support.sw.siemens.com to identify which version of Teamcenter the new MBSE Integration Gateway desktop client supports. Upgrade to the supported version of Teamcenter.
- Update to the supported version of Teamcenter.
- If upgrading from MBSE Integration Gateway version 4.3 to MBSE Integration Gateway version 5.0, delete the derby cache located in the *TC_ROOT\bhm\dataCacheDir\db\bhmCache* directory.

Procedure

1. Log on to Deployment Center and in the **Software** task.

2. In the **Software** task, add the version of the Teamcenter that you want to update to.

3. If you want to install new integrations, add the new integrations from the **Applications** task.

4. In the **Components** task, update any components if required.

   In the **MBSE Integration Gateway** component choose the **Enable Mass Client Deploy** option to deploy the MBSE Integration Gateway desktop client on multiple machines.

5. Once you have completed all the tasks you can deploy the environment on your desktop client.

Refer to the checklist **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** for next steps.

Step 6: Check if you need to perform optional client configurations for Teamcenter MBSE Integration Gateway client

You can perform the following optional configurations for Teamcenter MBSE Integration Gateway client:

Configuration	Description
Setup context and target folders	To specify the location of the context folder or parent folder and the target folder or the folder where the models are saved
Setup the cache directory	To specify the location of the derby cache directory
Enable data logging	To specify where to store the data logs
Enable support for multiple instances of Teamcenter MBSE Integration Gateway	To enable support for multiple instances of Teamcenter MBSE Integration Gateway clients
Enable the Active Workspace Open in Tool command	To enable the Open in Tool command on Active Workspace
Update the mmgenv.bat file with Teamcenter variable information	To specify Teamcenter variables required for integration

After you are done with performing these configurations, refer to the **Scenario 2: You already deployed Teamcenter MBSE Integration Gateway but want to update it and install new features** to see next steps.

Step 7: Test the integrations

Verify Enterprise Model Management

After deploying Enterprise Model Management, you can:

- Import Git data into Teamcenter

- Update Git data in Teamcenter
- Download Git data from Teamcenter

For more information on performing these tests, see *Enterprise Model Management in Active Workspace documentation*.

A. How the behavior modeling tool objects are represented in Teamcenter

Behavior modeling objects and relations

The following behavior modeling objects and relations are available once you install the behavior modeling integration:

Object name	Object type	Description
Behavior Model	Bhm0BehaviorModl	Represents a model from a modeling tool.
Additional Data	Relation	Represents additional data associated with the behavior model item.
Behavior Model Interface	Interface	Represents the abstract class that is the parent of the **Behavior Model Input Port** and **Behavior Model Output Port** classes.
Behavior Model Input Port	Bhm0InPort	Represents the input interface of the model.
Behavior Model Output Port	Bhm0OutPort	Represents the output interface of the model.
Behavior Model Connection	Bhm0Connection	Represents the connection between the ports of the models.
Behavior Model Component	Bhm0ModelComp	Represents model components. Model components are typically blocks or components that help define a model.
Behavior Model Subsystem	Bhm0Subsystem	Represents subsystems. Subsystems group elementary components to represent specialized behavior.
Behavior Model Elementary Components	Bhm0ElmModelComp	Represents elementary model components. The objects of this type are blocks that are not expected to be reused or have a lifecycle of their own.
Behavior Model Repository	Bhm0Repository	Represents a collection of model blocks.
Behavior Model Component Repository	Bhm0ComponentRep	Represents the library blocks from various modeling tools.

Object name	Object type	Description
Behavior Model Has Component	Bhm0HasComponent	Represents the relation with a reusable block and a library.
Behavior Model Associated Data	Bhm0AssociatedData	Represents the relation with a model and any data that is not primary or configuration data but still dependent on the model.

GT-POWER objects

The following GT-POWER objects and relations are available once you install the behavior modeling integration features:

Object name	Object type	Description
GTPower Model	Gtp0Model	Represents the GT-POWER model in Teamcenter.

System Modeling Workbench objects

The following System Modeling Workbench objects are available once you install the behavior modeling integration features:

Object name	Object type	Description
Capella Model	Sym0CapellaModel	Represents the Capella model in Teamcenter.
SysML Model	Sym0SysMLModel	Represents the SysML model in Teamcenter.

MagicDraw objects

The following MagicDraw objects and relations are available once you install the behavior modeling integration features:

Object name	Object type	Description
UML SysML Model	Uml0MLModel	Represents the MagicDraw UML SysML model in Teamcenter.
Magic Draw Model	Mdw0MDModel	Associates the MagicDraw model files to the UML SysML model or its subtypes.

Object name	Object type	Description
Rams Model	Ram0AnalysisModl	Represents the model used to perform reliability, availability, maintainability, and safety analysis using various tools such as MADe.
MADe Project	Mad0MADeModel	Associates the MADe Project file to the RAMS Analysis Model or its subtype.