SIEMENS

Teamcenter MBSE Integration Gateway 4.2

Deploying Teamcenter MBSE Integration Gateway

PLM00189 • 4.2

Contents

Chapter 1: Getting started with Teamcenter MBSE Integration Gateway

What is Teamcenter MBSE Integration Gateway?

Teamcenter MBSE Integration Gateway is an integration framework. It is used to integrate modeling tools such as MATLAB Simulink and Simcenter Amesim 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, 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 the modeling tool for model authoring and Teamcenter for model management. The integration helps leverage other Teamcenter benefits as follows:

- 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 framework provided by Teamcenter MBSE Integration Gateway is called the Model Management Gateway framework. This 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:

- Establish connections with Teamcenter, including SSO
- Define the data model and its deployment
- Define the mapping of external tool artifacts with Teamcenter business objects
- Manage file uploads and downloads

- Provide the user interface to support Teamcenter operations
- Manage customization and business object extensibility
- Maintain compatibility with multiple Teamcenter platforms

> **Note**
>
> If you are using Active Workspace in the *hosted* mode, the behavior modeling commands are hidden.

The Teamcenter MBSE Integration Gateway framework provides the following solutions:

- Model management
- System Analysis support

Understanding the Model Management Gateway framework

The Model Management Gateway is the framework that supports the integration of modeling tools with Teamcenter. The Gateway consists of the following components:

- Server
- Client

The server module consists of the SOAs, business logic, and data models. These data models can be extended for specific modeling tools.

The client module consists of Java APIs, extensions, and connectors. The client can be Rich Client or Active Workspace. The Java APIs interact with Teamcenter SOAs.

The connector resides in the modeling tools and exchanges data between the modeling tool and Teamcenter. The data that is exchanged is mapped using a file called the integration definition file. This is an XML file that specifies:

- Mapping of modeling tool objects to Teamcenter objects
- Mapping of object attributes
- Mapping of file extensions
- Organization data that is used to configure folders

If you are using the common connector-based integration, you need not specify the mapping in the integration definition file.

Understanding System Analysis

System Analysis includes behavioral, functional, and failure mode analysis of systems using analysis request.

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 a limited integration of the modeling tool with Teamcenter. In this mode, all the model files 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.

Chapter 2: Installing Teamcenter MBSE Integration Gateway

Overview of installing Teamcenter MBSE Integration Gateway

You can install Teamcenter MBSE Integration Gateway as follows:

- Install a Teamcenter version that supports MBSE Integration Gateway.

  For a list of Teamcenter versions that support MBSE Integration Gateway, refer to the Integrations Matrix on the Hardware and Software Certifications page on GTAC.

- Install Active Workspace.

- If this is the first time you are installing MBSE Integration Gateway, do the following:
  - Install a new MBSE Integration Gateway patch on a Teamcenter platform.
  - For Simulink integration, install the MBSE Integration Gateway server and client features on a 4-tier or 2-tier Teamcenter server and client.
  - For Simcenter Amesim integration, install the MBSE Integration Gateway server features.

    Simcenter Amesim integration requires Active Workspace as well as Simcenter Amesim configurations.
  - Analysis Request Programming Interfaces are used by the Simcenter System Synthesis tool. You must also install the Simcenter System Synthesis server components.

- If you already have MBSE Integration Gateway, you can update your existing Teamcenter MBSE Integration Gateway to a new version.

- If you only need to push changes from your modeling tool to the Teamcenter server and do not need the rich client features on your machine, you can install only the simulink integration your machine.

When applying patches, the order of applying patches matters. First apply the Teamcenter patch, then the Active Workspace patch, and then the MBSE Integration Gateway patch.

For example, if your existing environment consists of Teamcenter 11.5, Active Workspace 4.0, and MBSE Integration Gateway 4.0 and you want to upgrade to Teamcenter 11.5.0.5, Active Workspace 4.0.3, and MBSE Integration Gateway 4.0.4, apply the patch in the following order:

Teamcenter > Active Workspace > MBSE Integration Gateway

Installing Teamcenter MBSE Integration Gateway on a Teamcenter server

Install MBSE Integration Gateway on a supported Teamcenter environment

Before installing the MBSE Integration Gateway features, you must overlay MBSE Integration Gateway to a supported Teamcenter environment.

1. From GTAC, download and extract the contents of the MBSE Integration Gateway patch zip file **MBSE Integration Gateway***release*_**Tc***rel-num_platform*.**zip** to any folder.

2. If any server managers are running on the corporate server to be patched, shut them down before proceeding with the patch.

3. Launch Teamcenter Environment Manager from the *TC_ROOT*/**install** folder.

   For example, if your base Teamcenter platform is 12.2, launch Teamcenter Environment Manager from the 12.2 *TC_ROOT*/**install** directory.

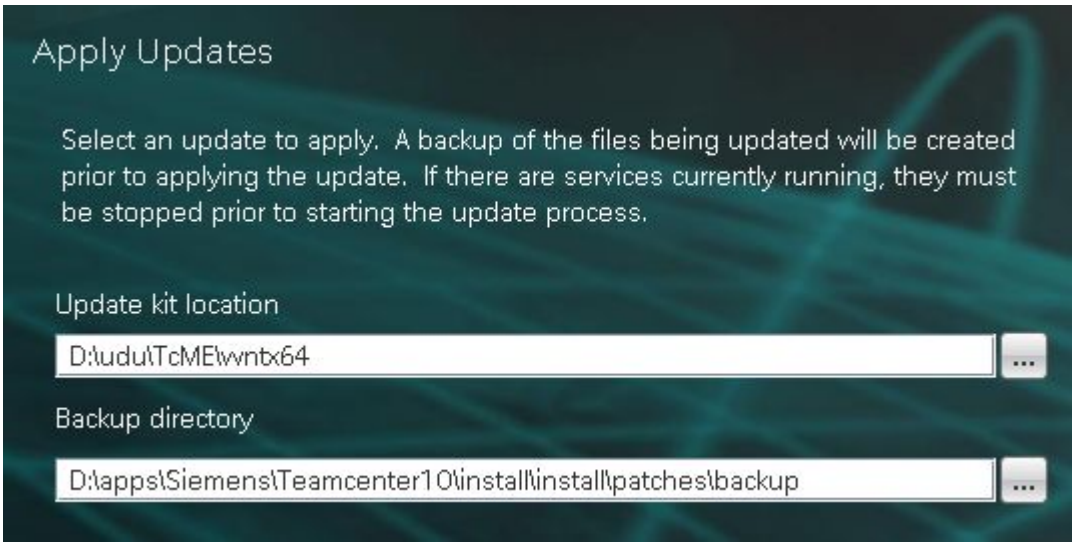
4. In the **Media Locations** panel, specify the locations of the base Teamcenter and the MBSE Integration Gateway install files and click **Next**.

5. In the **Maintenance** panel, select **Updates Manager** and click **Next**.

6. In the **Apply Updates** panel:

   - In the **Update kit location** box, type or browse to the location where you extracted the contents of **MBSE Integration Gateway***release*_**Tc***rel-num_platform*.**zip**.

   - In the **Backup directory** box, type or browse to the location where you wish to create the backup files and click **Next**.



7. The **Status Message** window informs you that TEM will stop any running Teamcenter services in order to perform the update. Once the update is completed, these services will be restarted. Any users currently on the system will be logged out.

   Click **Close**.

8. In the **Teamcenter Administrative User** panel, type the user password and click **Next**.

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

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

Install Teamcenter MBSE Integration Gateway features on a 2-tier or 4-tier Teamcenter server

You can separately install the server and client components of Behavior Modeling. The following steps describe how to install the behavior model server and client components on the same machine.

1. Launch the Teamcenter Environment Manager (TEM) for the corporate server to which the MBSE Integration Gateway features are to be added.

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 the corporate server configuration and click **Next**.

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

6. To enable concurrent modeling on Active Workspace server:

   - In the **Features** panel, expand **Active Workspace→Server Extensions** and select **Concurrent Modeling**.

7. In the **Features** panel, expand **Extensions→Model Management** and select the MBSE Integration Gateway features you wish to include.

8. To enable model management features for Active Workspace, under **Model Management**, expand **Active Workspace** and select::

   - **Branch Console** to install concurrent modeling features.
   - **System Modeling Integration** for integration with Systems Modeling.
   - **Open in Tool** to open model files from Active Workspace.

9. To install the server components on a server, under **Model Management**, expand **Server** and select:

   - **Branch Data Organization** to install concurrent modeling.
   - **Gateway for modeling** to install the common behavior modeling framework.
   - **Branching and Versioning Foundation** to install the support for concurrent modeling.
   - **GTPower Integration** to install the GTPower data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **LMS Amesim Integration** to install the Simcenter Amesim data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **MADe Integration** to install the MADe data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **Magic Draw Integration** to install the Magic Draw UML SysML data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **Simulink Integration** to install the Simulink data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **System Modeling Workbench Integration** to install the System Modeling Workbench data model.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **Support for Concurrent Modeling** to install concurrent modeling features.

     To install this option, you must also choose the **Gateway for modeling** option.
   - **System Synthesis Modeling** to install the Simcenter System Synthesis data model.

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

10. To install **Analysis Request Programming Interface**, under **Extensions**, expand **Systems Driven Product Development** and select **Analysis Request Programming Interface**.

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

    This informs you that all Teamcenter services and processes (with the exception of any FSCs) must be shut down before continuing. Shut down these services and processes and click **OK**.

12. In the **Teamcenter Administrative User** panel, type the user password and click **Next**.

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

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

Update the Teamcenter MBSE Integration Gateway patch

To update the MBSE Integration Gateway patch to a supported Teamcenter environment:

1. From GTAC, download and extract the contents of the MBSE Integration Gateway patch zip file **MBSE Integration Gateway***release*_**Tc***rel-num_platform***.zip** to any folder.

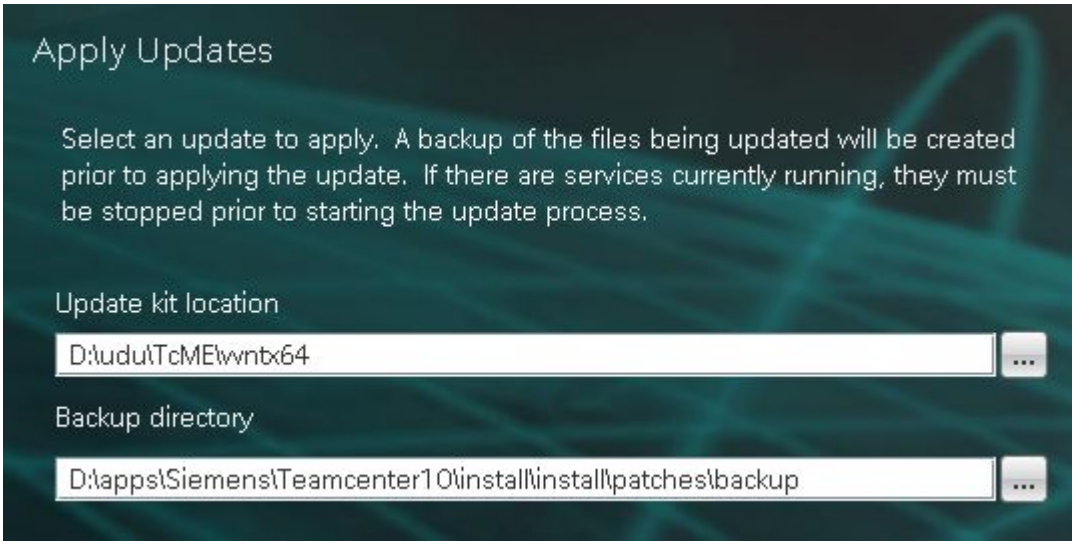
2. If any server managers are running on the corporate server to be patched, shut them down before proceeding with the patch.

3. In the **Media Locations** panel, specify the locations of the base Teamcenter and the MBSE Integration Gateway install files.

4. Click **Next**.

5. In the **Maintenance** panel, select **Updates Manager** and click **Next**.

6. In the **Apply Updates** panel:
   - In the **Update kit location** box, type or browse to the location where you extracted the contents of **MBSE Integration Gateway***release*_**Tc***rel-num_platform*.**zip**.
   - In the **Backup directory** box, type or browse to the location where you wish to create the backup files.
   - Click **Next**.



7. Click **Close** to close the **Status Message** window.
8. The **Optional Configuration Enhancements** panel informs you of the enhancements that can be installed.
   a. Click **View Enhancement Info** for more information about the enhancements.
   b. Select **Yes** to install the enhancements.
9. In the **Teamcenter Administrative User** panel, type the user password and click **Next**.
10. In the **Confirmation** panel, click **Start**.
11. If you receive a message that the *TC_DATA* directory is not updated, you must manually update the directory as follows:
    a. Create backups of your current *TC_DATA* directories.
    b. Extract the **platform\tc\data.zip** file from the temporary location you created in step 1.
    c. Copy the extracted contents of the **data.zip** file to your *TC_DATA* directories, overwriting existing files.
12. When the update is complete, click **Close** to close TEM.

Installing Teamcenter MBSE Integration Gateway on a local machine

Install the Teamcenter MBSE Integration Gateway client

1. From GTAC, download and extract Teamcenter platform kits to a temporary location. 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.

2. From GTAC, download and extract the contents of the MBSE Integration Gateway patch zip file **MBSE Integration Gateway***release*_**Tc***rel-num_platform*.**zip** to any folder.

3. Launch Teamcenter Environment Manager from the Teamcenter patch kit. For example, launch Teamcenter Environment Manager from the 10.1.4 **install** directory.

4. In the **Welcome to Teamcenter** panel, click **Install**.

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

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

   b. Teamcenter patch location install file location in the **Update Location** list.

   c. The MBSE Integration Gateway install file location in the **Update Location** list.

6. Click **Next**.

7. Skip the **Solutions** panel.

8. In the **Features** panel, expand **Extensions→Model Management**, expand **Client**, and select:.

   - **Gateway for modeling** to install the common framework components on the client.

- **LMS System Synthesis Modeling** to install the Simcenter System Synthesis integration connector.

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

- **Simulink Integration** to install the MATLAB integration connector.

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

- **Support for Concurrent Modeling** to install concurrent modeling support.

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

9. Click **Next**.

10. 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.

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

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

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

Install only the Behavior Modeling client for Simcenter Amesim and Simcenter System Synthesis integration

Simcenter Amesim and SystemSynthesis integration requires a 32-bit Teamcenter MBSE Integration Gateway client. Before you install Simcenter Amesim and Simcenter SystemSynthesis integration:

- Ensure that the correct JRE environment variables are set as follows:
  - The **JRE_HOME** variable should point to a 32-bit JRE installation.
  - The **JRE64_HOME** variable should point to a 64-bit JRE installation.
- Ensure that you download the correct kits.

To install Simcenter Amesim and Simcenter SystemSynthesis integration on a 32-bit Teamcenter MBSE Integration Gateway client, do the following:

1. From GTAC, download and extract the appropriate Teamcenter platform kits to a temporary location. The downloaded kits must match the version of the Teamcenter server. For example, if the Teamcenter server version is 11.2, download the Teamcenter 11.2 and 11.2.2 kits.

2. From GTAC, download and extract the contents of the appropriate MBSE Integration Gateway patch zip file (**MBSE Integration Gateway***release*_**Tc***rel-num_platform*.**zip**) to any folder.

   Ensure that the MBSE Integration Gateway kit you download is 32-bit.

3. Launch Teamcenter Environment Manager (TEM) from the Teamcenter patch kit. For example, launch TEM from the **install** directory.

4. In the **Welcome to Teamcenter** panel, click **Teamcenter**.
5. In the **Install/Upgrade Options** panel, click **Install**.
6. In the **Media Locations** panel:
   a. Specify the locations of the base Teamcenter install in the **Original Medial Location** box.
   b. Specify the Teamcenter patch location and the 32-bit MBSE Integration Gateway install file location in the **Update Location** list
   c. Click **Next**.

7. In the **Features** panel:
   - Expand **Extensions→Model Management→Client** and select **Gateway for Modeling**.
   - Expand **Extensions→Systems Driven Product Development** and select **Analysis Request Programming Interface**.

     Ensure that the Analysis Request functionality is also deployed on the Teamcenter server.
8. Click **Next** till you reach the **Teamcenter Mechatronics Engineering Common Client** panel.
9. In the **Teamcenter Mechatronics Engineering Common Client** panel, type the connection URL.
10. In the **Confirmation** panel, click **Start**.

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

Install Teamcenter MBSE Integration Gateway features using Deployment Center

This topic documents how to install Teamcenter MBSE Integration Gateway using Deployment Center. For detailed information about how to use Deployment Center, see the Deployment Center help collection.

Using Deployment Center you can install server and client components in the same distributed environment by providing their machine names. Deployment Center generates separate scripts for each machine.

For example, you require **Corporate Server**, **Active Workspace**, and **RAC 4-Tier** with **Simulink Integration**. To meet this goal, you can install **Corporate Server** and **Active Workspace** on machine A, and **RAC 4-Tier** with **Simulink Integration** on machine B.

This creates deploy scripts for both machines, which you can then deploy on each machine.

1. Log on to Deployment Center.
2. In the **Applications** task, based on your requirements, select the following applications from the **Available Applications** list:

Application	Description
To install concurrent modeling features:	
Teamcenter→Active Workspace→Branching and Versioning	Installs the concurrent modeling functionality
Teamcenter→Foundation→Model Management→Branch Data Organization	Installs concurrent modeling
Teamcenter→Foundation→Model Management→Branching and Versioning Foundation	Installs support for concurrent modeling
To install support for Simcenter System Synthesis:	
Teamcenter→Foundation→Model Management→LMS System Synthesis Modeling	Installs the Simcenter System Synthesis data model
To install Teamcenter MBSE Integration Gateway features for Active Workspace:	
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Active Workspace→Branch Console	Installs concurrent modeling features
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Active Workspace→Open in Tool	Opens model files from Active Workspace
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Active Workspace→System Modeling Integration	Provides integration with System Modeling

Application	Description
To install model management features on a local client machine:	
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Client→Gateway for modeling	Installs the common framework components on the client
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Client→LMS System Synthesis Modeling	Installs the Simcenter System Synthesis integration connector
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Client→Simulink Integration	Installs the MATLAB integration connector
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Client→Support for Concurrent Modeling	Installs concurrent modeling support
To install model management features on the Teamcenter server:	
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→GT Power Integration	Installs the GTPower data model
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→Gateway for modeling	Installs the model management framework
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→LMS Amesim Integration	Installs the Simcenter Amesim data model
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→MADe Integration	Installs the MADe data model
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→Magic Draw UML SysML modeling	Installs the Magic Draw UML SysML data model
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→Simulink Integration	Installs the Simulink data model
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→Support for Concurrent Modeling	Installs support for concurrent modeling
Teamcenter→Foundation→MBSE Integration Gateway→Model Management Server→System Modeling Workbench Integration	Installs the System Modeling Workbench data model
To install Verification and Validation APIs:	

Application	Description
Teamcenter→Foundation→Systems Driven Product Development→Verification and Validation Programming Interface	Installs Verification and Validation APIs

3. To update MATLAB configuration information, from the **Components** tab, select the **RAC 2-Tier** or the **RAC 4-Tier** component, and in the **MATLAB Client Information** section, do the following:

   a. In the **MATLAB Installation Directory** box, type the location where MATLAB is installed. This is the directory containing the **bin** folder.

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

4. To update Simcenter Amesim and Simcenter System Synthesis integration configuration information, from the **Components** tab, select the **RAC 2-Tier** or **RAC 4-Tier** component, and in the **MBSE Integration Gateway Common Client** section, type the connection URL.

5. Generate deployment scripts.

Chapter 3: Configuring Teamcenter MBSE Integration Gateway

Performing server-side configurations for Teamcenter MBSE Integration Gateway

Mapping the data using the integration definition file

Data mapping using the integration definition file

You can map Teamcenter business objects to the artifacts in the external tool, using the integration definition file.

- To ensure that you can reuse the artifacts in the external tool, you must map these artifacts to items or the items' subtypes.
- Use GDE elements or their subtypes to map to other elements such as traceability.
- You can associate the following behavior types between a container and its contents:
  - **BVR**

    Represents heirarchical or structure data. This behavior is used for artifacts such as Simulink models that are independent entities. When such models are saved to Teamcenter, an occurence model with the top line as the container artifact object and child lines corresponding to the component objects of that container are created in Teamcenter.

  - **GRM**

    Represents associations or relations (equivalent to aggregation). This behavior is defined for artifacts such as Simulink libraries that are containers of various reusable components. A relationship is established between the container and its components, using the GRM object and its corresponding rules.

The data mapping in the integration definition file is defined as follows:

- **ObjectMapping**

  Used for mapping to the top-level elements or containers such as Simulink models or libraries.

  ```
  <ObjectMapping type="Model" behaviorType="BVR" tctype="Bhm0BehaviorModl">
  ```

  The **ObjectMapping** element contains the following attributes:

  - **type**

    Represents the top-level element or container of the external tool, such as a model or a library.

  - **behaviorType**

    Represents how the container is saved in Teamcenter. Use either **BVR** or **GRM**.

Use **BVR** to represent heirarchical data such as models, and use **GRM** to represent relational data such as libraries.

- **tctype**

  Represents the Teamcenter object that maps to the container object of the external tool.

- **BHMElement**

  Represents the components of a specific container.

  ```
  <BHMElement type="RootModel" tctype="Bhm0BehaviorModl">
  ```

  It contains the following attributes:

  - **type**

    Represents the object type of the external tool.

    By default, the **type** of the top-level model or the base container is **RootModel**.

  - **tctype**

    Represents the Teamcenter object that maps to the container object of the external tool.

  - **reftype**

    Represents the root model or the base type of the container.

  You can specify the attribute mappings in the **AttributeMappings** section of the **BHMElement**.

  > **Note**
  >
  > All components of a container support only hierarchical (BVR) behavior. For example, the contents of a SubSystem block type of MATLAB Simulink are saved as BVR although the container is configured as GRM.

Mapping Simcenter System Synthesis data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of Simcenter System Synthesis, using the Simcenter System Synthesis integration definition file. The file is named **SSM_TCMEIntegrationDefinition.xml**, and its format is similar to the format of the integration definition file.

The data mapping in the Simcenter System Synthesis integration definition file is defined using the following elements:

- **ObjectMapping**

  Used for mapping to the top-level elements or containers such as AbstractSynthesis.

  ```
  <ObjectMapping type="AbstractSynthesis">
  ```

  The **ObjectMapping** element contains the following attribute:

  - **type**

Represents the top-level element or container of Simcenter System Synthesis, such as AbstractSynthesis.

- **SSMElement**

  Defines the mappings between Simcenter System Synthesis and Teamcenter objects.

```
<SSMElement type="Dictionary" tctype="Ssm0SimDictionary">
    <AttributeMappings>
      <AttributeMapping  tcattr="object_name" name="Name"/>
      <AttributeMapping  tcattr="object_desc" name="Desc"/>
    </AttributeMappings>
 </SSMElement>
```

  The **SSMElement** contains the following attributes:

  o **type**

    Represents the object type of Simcenter System Synthesis, for example, **Dictionary**.

  o **tctype**

    Represents the Teamcenter object type.

  You can specify how the attributes map in the **AttributeMappings** section of the **SSMElement**. For example, you can map attributes of the **Dictionary** element to Teamcenter properties by updating the following attributes:

  o **tcattr**

    Specifies the name of the Teamcenter property.

  o **name**

    Specifies the name of the Simcenter System Synthesis attribute.

```
<SSMElement type="Dictionary" tctype="Ssm0SimDictionary">
    <AttributeMappings>
      <AttributeMapping  tcattr="object_name" name="Name"/>
      <AttributeMapping  tcattr="object_desc" name="Desc"/>
    </AttributeMappings>
 </SSMElement>
```

  If you have a custom object, the required properties for the inputs must be defined in the **Attribute Mappings** section.

- **FileMap**

  Defines the mapping between Teamcenter datasets and Simcenter System Synthesis datasets.

```
<FileMapping>
  <FileMap fileExt="zip" tcDatasetType="Ssm0SimDictionaryDataset"
```

```
    type="DictionaryDataset" tcNameReferencedType="Ssm0SimFileReference" />
    <FileMap fileExt="zip" tcDatasetType="Ssm0SimTemplateDataset"
    type="TemplateLibraryDataset" tcNameReferencedType="Ssm0SimFileReference" />
</FileMapping>
```

The **FileMap** element contains the following attributes:

- **fileExt**

  Specifies the file extension of the dataset, for example, **zip**. You can find the value of the extension from the **References** tab of the dataset business object in Business Modeler IDE.

- **tcDatasetType**

  Specifies the Teamcenter dataset type.

- **type**

  Specifies the Simcenter System Synthesis file type.

- **tcNameReferencedType**

  Specifies the named reference type. You can find the value of the named reference from the **References** tab of the dataset business object in Business Modeler IDE.

- **RelationMapping**

Specifies the relations to create between primary and secondary objects when creating or updating Simcenter System Synthesis objects in Teamcenter.

```
<RelationMapping primaryType = "Ssm0SimModel"
secondaryType="Bhm0ModelCompRevision" relationType="Ssm0ModelCompRelation"/>
```

The **RelationMapping** element contains the following attributes:

- **primaryType**

  Specifies the Teamcenter primary object type.

- **secondaryType**

  Specifies the Teamcenter secondary object type.

- **relationType**

  Specifies the Teamcenter relation type.

> **Note**
>
> When creating relations between Simcenter System Synthesis objects and Simcenter Amesim or MATLAB objects, the attribute **secondaryType** must match the Teamcenter business object type used for Simcenter Amesim or MATLAB integration.
>
> For example, if MATLAB uses the **Bhm0SubSystem** type for SubSystem, the secondary type must be **Bhm0SubSystemRevision**.

Update the Simcenter System Synthesis integration definition file

1. In Teamcenter, search for **SSM_TCMEIntegrationDefinition**.
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 Simcenter System Synthesis integration client to use the updated configuration.

Map Simcenter Amesim data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of Simcenter Amesim by updating the Simcenter Amesim integration definition file. The file is named **AMESIM_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 these objects are mandatory, you must map the attributes to Simcenter Amesim objects using the integration definition file.

To update the integration definition file:

1. In Teamcenter, search for **AMESIM_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 Simcenter Amesim integration client to use the updated configuration.

For information about configuring the Simcenter Amesim integration with Teamcenter, see the Simcenter Amesim help collection.

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 Magic Draw 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 GT-POWER data to Teamcenter using the integration definition file

You can map Teamcenter business objects to the artifacts of GT-POWER by updating the GT-POWER integration definition file. The file is named **GTPOWER_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 GT-POWER objects using the integration definition file.

To update the integration definition file:

1. In Teamcenter, search for **GTPOWER_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 GT-POWER 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.

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 moeling 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>
```

Enable the Active Workspace Open in Tool command

To enable the opening of modeling tools from Active Workspace, perform the following Teamcenter server and client configurations:

Teamcenter server configurations

- Using **Updates Manager** in Teamcenter Environment Manager, apply the Teamcenter MBSE Integration Gateway patch.
- Open the **web.xml** file located in the *TC_ROOT*/**aws2/stage/repo/kit/tc-aw-solution/src_j2ee/WEB-INF** folder.
- Add the following entries to the **servlet** section of the **web.xml** file:

```
<servlet>
  <servlet-name>TcMELauncherServlet</servlet-name>
  <servlet-class>com.siemens.splm.clientfx.tcmeproxies.server.TCMELauncherServlet
</servlet-class>
</servlet>
```

- Add the following to the **servlet-mapping** section of the **web.xml** file:

```
<servlet-mapping>
  <servlet-name>TcMELauncherServlet</servlet-name>
  <url-pattern>/tcmelauncher/*</url-pattern>
</servlet-mapping>
```

- Save the changes made to the **web.xml** file.
- Start Teamcenter Environment Manager, and select the **Active Workspace→Client→Open in Tool** option. This generates the Active Workspace WAR file (**awc.war**) again.
- Deploy this Active Workspace WAR file to a web server.

If a .NET server is used, perform the following steps:

1. Open the **Web.config** file located in the *TC_ROOT*/**aws2/stage/repo/kit/tc-aw-solution/src_iis** folder.
2. Add the following entries to the **Web.config** file:

```
<location path="tcmelauncher/fnd0branch">
  <system.webServer>
    <handlers>
      <add name="TcmeLauncherhandler" verb="*" path="*"
      type="com.siemens.splm.clientfx.tcmenetproxies.Launcher.TcmeLauncherhandler,
      TcmeLauncherHandler" />
    </handlers>
  </system.webServer>
</location>

<location path="tcmelauncher/Mat0Model">
  <system.webServer>
    <handlers>
      <add name="TcmeLauncherhandler" verb="*" path="*" type="com.siemens.splm.
      clientfx.tcmenetproxies.Launcher.TcmeLauncherhandler, TcmeLauncherHandler" />
    </handlers>
  </system.webServer>
</location>
```

3. Save the changes to the **Web.config** file.
4. Start Teamcenter Environment Manager and select the **Active Workspace→Client→Open in Tool** option. This generates the Active Workspace WAR file (**awc.war**) again.
5. Deploy this Active Workspace WAR file to a web server.

Client configurations

After clicking the downloaded file in the browser, define the following registry entries:

The examples described here are specific to the Simcenter Amesim tool.

1. Define the MIME Type. This specifies the dataset name corresponding to each tool. For example, for Simcenter Amesim, use **il0amesim**.

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application\il0amesim]
"Extension"=".tcamexml"
```

2. Define the command to be executed. The key must start with the dataset name of the file.

   The value of the command is the executable, batch, or script to be executed when you click the downloaded file.

```
[HKEY_CLASSES_ROOT\il0amesimfile]
[HKEY_CLASSES_ROOT\il0amesimfile\shell]
[HKEY_CLASSES_ROOT\il0amesimfile\shell\open]
@="&open"
[HKEY_CLASSES_ROOT\ il0amesimfile \shell\open\command]
@="<Path of AMESIM executable>\\amesim.exe\"%1\""
```

3. Map the file extension to the MIME content type.

```
[HKEY_CLASSES_ROOT\.tcamexml]
"Content Type"="application/il0amesim"
@="il0amesimfile"
```

Alternatively, you can import the registry changes through the **regedit** application as follows:

1. Create a file and provide a name, such as **AMESimLauncher.reg**.

2. Paste the following lines to the file and save the **AMESimLauncher.reg** file:

```
Windows Registry Editor Version 5.00
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application\il0amesim]
"Extension"=".tcamexml"

[HKEY_CLASSES_ROOT\il0amesimfile]
[HKEY_CLASSES_ROOT\il0amesimfile\shell]
[HKEY_CLASSES_ROOT\il0amesimfile\shell\open]
@="&open"

[HKEY_CLASSES_ROOT\ il0amesimfile\shell\open\command]
@="<Path of AMESIM executable>\\amesim.exe\"%1\""

[HKEY_CLASSES_ROOT\.tcamexml]
"Content Type"="application/il0amesim"
@="il0amesimfile"
```

3. Start the regedit application and import the **AMESimLauncher.reg** file.

Configure an analysis request

1. **Create the analysis definition**

   Each analysis 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 that are permitted in an analysis 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 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 Analysis 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 Analysis 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 analysis request mapping**

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

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

5. **Update the analysis request stylesheet**

   Update the analysis 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
```

Classifying 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 *Classification Admin* in the Teamcenter help collection.

  For information about classifying models, see *Classification* in the Teamcenter help collection.

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

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 *Configure your business data model in BMIDE* in the Teamcenter help.

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

For more information, see *Specifying additional data mapping by modifying the integration definition file*.

Performing client-side configurations for Teamcenter MBSE Integration Gateway

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.

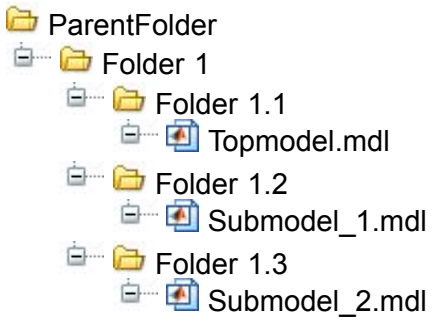
  Example

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

The location where your model files are downloaded are 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 following directory structure **D:\models\ContextExample\Level1\Level2\Level3\Level4** and your staging directory is located in **D:\staging** . The following table shows 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**

Set up 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*.

Enable data logging

Teamcenter MBSE Integration Gateway uses the log4j mechanism for 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.

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 with 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"
```

Enable the branch console by updating registry entries

Add the following entries to the Windows registry to enable branch console:

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/fnd0branch]
"Extension"=".bnv"

[HKEY_CLASSES_ROOT\.bnv]
"Content Type"="application/fnd0branch"
@="fnd0branchfile"

[HKEY_CLASSES_ROOT\fnd0branchfile]

[HKEY_CLASSES_ROOT\fnd0branchfile\shell]

[HKEY_CLASSES_ROOT\fnd0branchfile\shell\open]
@="&open"

[HKEY_CLASSES_ROOT\fnd0branchfile\shell\open\command]
@="<TC_ROOT>\\bhm\\openInConsole.bat \"%1\""
```

Add the following entries to the Windows registry to enable the merge functionality:

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/fnd0branchMerge]
"Extension"=".mrg"

[HKEY_CLASSES_ROOT\.mrg]
"Content Type"="application/fnd0branchMerge"
@="fnd0branchfileMerge"


[HKEY_CLASSES_ROOT\fnd0branchfileMerge]

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell]

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell\open]
@="&open"

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell\open\command]
@="<TC_ROOT>\\bhm\\mergeBranchNodes.bat \"%1\""
```

Enable the Active Workspace Open in Tool command

To enable the opening of modeling tools from Active Workspace, perform the following Teamcenter client configurations:

1. After clicking the downloaded file in the browser, define the following registry entries:

   The examples described here are specific to the Simcenter Amesim tool.

   Define the MIME Type. This specifies the dataset name corresponding to each tool. For example, for Simcenter Amesim, use **il0amesim**.

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application\il0amesim]
"Extension"=".tcamexml"
```

2. Define the command to be executed. The key must start with the dataset name of the file.

   The value of the command is the executable, batch, or script to be executed when you click the downloaded file.

```
[HKEY_CLASSES_ROOT\il0amesimfile]
[HKEY_CLASSES_ROOT\il0amesimfile\shell]
[HKEY_CLASSES_ROOT\il0amesimfile\shell\open]
@="&open"
[HKEY_CLASSES_ROOT\ il0amesimfile \shell\open\command]
@="<Path of AMESIM executable>\\amesim.exe\"%1\""
```

3. Map the file extension to the MIME content type.

```
[HKEY_CLASSES_ROOT\.tcamexml]
"Content Type"="application/il0amesim"
@="il0amesimfile"
```

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
```

Chapter 4: Configure MATLAB Simulink integration

Performing server-side configurations for MATLAB Simulink integration

Update MATLAB Simulink integration with a new version of MATLAB Simulink

To update the Teamcenter MATLAB Simulink integration with a new version of MATLAB Simulink:

1. Delete the **classpath.txt** file located in the *TC_ROOT*/**bhm/scripts** directory, if the file exists.
2. Delete the **javaclasspath.txt** file located in the *TC_ROOT*/**bhm/scripts** directory, if the file exists.
3. Update **MATLAB_ROOT** in the following files:
   - *TC_ROOT*/**bhm/matlab/scripts/start_matlab.bat**
   - *TC_ROOT*/**bhm/matlab/load.bat**

     When you upgrade MATLAB Simulink from a 32-bit to a 64-bit version, the **MATLAB_ROOT** variable must point to the 64-bit version.

Set the path variable

Update the **PATH** variable pointing to the **tcs/lib** and the bhm matlab library in the following files:

- *TC_ROOT*/**bhm/matlab/scripts/start_matlab.bat**
- *TC_ROOT*/**bhm/matlab/load.bat**

  When you upgrade MATLAB Simulink from a 32-bit to a 64-bit version, the **MATLAB_ROOT** variable must point to the 64-bit version.

Example

```
set PATH=%MATLAB_ROOT%\lib;%TC_ROOT%;D:\Program Files\Siemens\
Teamcenter11\tccs\lib;D:\Program Files\Siemens\Teamcenter11\bhm\matlab\
native\wnti\lib;%PATH%
```

Configure the opening of models in a specific version of MATLAB Simulink

To open MATLAB Simulink in integration mode, ensure that the user has administration privileges.

To open a model in a specific version of MATLAB Simulink from the Teamcenter rich client, the user must have administration privileges.

You must also update the **load.bat** file located in *TC_ROOT*/**bhm/matlab** with the following command:

```
call "%MATLAB_ROOT%\bin\matlab.exe" -regserver -nojvm -nosplash -nodesktop
 -r "exit"
```

Alternatively, you can register the MATLAB Simulink version as an Automation Server in Windows before explicitly triggering the open operation by running the command described previously.

For registering MATLAB as an Automation Server in Windows, see the MATLAB help documentation from Mathworks.

Specifying additional data mapping by modifying the integration definition file

Teamcenter provides an integration definition file, which maps the modeling tool objects, attributes, and file types to the corresponding Teamcenter objects.

> **Caution**
>
> Modifying the integration definition file changes the behavior of the integration. Proceed with extreme caution.

The name of the integration definition file is *TOOL NAME_* **BHMIntegrationDefinition.xml**, for example, **MATLAB_BHM_INT_DEF_FILE.xml**.

Teamcenter downloads this file to a temporary location during runtime. The file is created in the Teamcenter database during installation. Only Teamcenter administrators can modify this file.

For more information about modifying the file, see *Modify the integration definition file*.

Configure how to save model folders

To save the folder structure of the model when saving the model to Teamcenter, update the **OrganizationData** section of the **MATLAB_BHM_INT_DEF_FILE.xml** as follows:

```
<OrganizationData>
  <Folder name="MODELFOLDER" tcRelation=""/>
  <Folder name="const" tcRelation="IMAN_requirement"/>
  <Folder name="image" tcRelation="IMAN_reference"/>
  <Folder name="extern" tcRelation="IMAN_reference"/>
  <Folder name="simdata" tcRelation="IMAN_requirement"/>
</OrganizationData>
```

- **MODELFOLDER** is a keyword that identifies any folder containing model files corresponding to the models to be managed in Teamcenter.
- **tcRelation** specifies the relation with which the files from the corresponding folders are associated with the model object.
- **Folder name** specifies the folders whose contents must be saved to Teamcenter and attached to the model object, using the relation specified by the **tcRelation** attribute.

For example, **const**, **image**, **extern**, and **simdata** refer to folders whose content must be saved along with the model.

- If you do not specify any relation for the folders, the default relation **IMAN_Reference** is used.

Siemens PLM Software recommends that you not use the XML structure with the **MODELFOLDER** element if the top model and all its submodels are organized in the same folder. If you use the **MODELFOLDER** element, the model files are also associated as configuration data.

For configuring folder image files, such as .jpg or .png files, avoid using the relations **Thumbnail**, **IMAN_manifestation**, **IMAN_specification**, **IMAN_Rendering**, **IMAN_Motion**, or **IMAN_3D_snap_shot**.

If the configured image files are associated with the relations described previously, the preview image of the model is not consistent.

Modify the integration definition file

Caution

Modifying the integration definition file changes the behavior of the integration. Proceed with extreme caution.

Note

You must update the integration definition file when you upgrade to a new version of Teamcenter. Teamcenter does not automatically update this file.

Update the original integration file with the new entries mentioned in the new integration definition file.

In the Teamcenter MBSE Integration Gateway install kit, this file is available in the *Install_ROOT*/**tc/matlabint_install.zip** directory.

Teamcenter provides an integration definition file, which maps the modeling tool objects, attributes, and file types to the corresponding Teamcenter objects.

The name of the integration definition file is *TOOL NAME*_ **BHMIntegrationDefinition.xml**, for example, **MATLAB_BHM_INT_DEF_FILE.xml**.

Teamcenter downloads this file to a temporary location during runtime. The file is created in the Teamcenter database during installation. Only Teamcenter administrators can modify this file.

To modify this file:

1. From My Teamcenter, search for the dataset **MATLAB_BHM_INT_DEF_FILE**.
2. Right-click the integration definition file and select **Named References**

   OR

   From My Teamcenter, choose **View→Named References**.

3. In the **Named References** dialog box, select the integration definition file, and click the **Download** button.
4. Export the file, make the desired changes, and save the file.
5. In the **Named References** dialog box, click the ✕ button to delete the existing file.
6. Click the **Upload** button and import the modified integration definition file.

   Caution

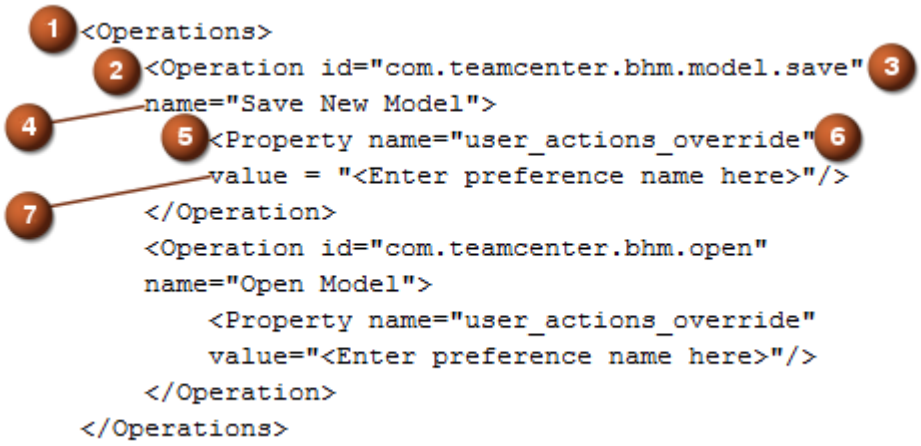
   Do not change the name of the file.

7. Click **Close** to close the **Named References** dialog box.

Override the default behavior of the integration dialog boxes

You can override the default behavior of the integration dialog boxes. For example, you can disable the user from navigating the **Home** folder during the save operation.

To specify the override actions, modify the integration definition file as follows:



1	Create an **Operations** section that acts as a container for all overridden behaviors.
2	For each integration operation you want to modify, create an **Operation** section. The **Operation** section has two attributes, namely, the **Operation Id** and **name**.
3	The **Operation Id** specifies the unique identifier of the operation, for example, **com.teamcenter.bhm.model.save**, for saving a new model.
4	In the **name** attribute, specify the name for the operation.
5	In the **Operations** section, add a **Property** element that contains two attributes.
6	The **name** attribute specifies that the property is used to override actions. Its value must always be **user_actions_override**.
7	The **Value** attribute specifies the name of a preference. This preference denotes an action that must be overridden. For example, the **NAVIGATE_CLASSIFICATION_TREE** preference denotes if a user can navigate the classification tree. These preferences are not available by default and must be created by the administrator.
8	These preferences can have one of the following values: • **NEVER** Specifies that the action must never be allowed. The user interface entries for the action will be disabled. • **ALWAYS** Specifies that the action must be allowed. The user interface entries for the action will be intialized with the desired state and then disabled. • **DEFAULT** Specifies the default value on the user interface corresponding to the user action. The user interface remains enabled so that the default values can be changed. The default value may not be applicable to all user actions.

You can override the following user actions:

- Save a new model
- Overwrite an existing model

- Open a model
- Insert a model
- Check out a model
- Open a model from the Rich Client

Operation names and preferences to override the default behavior of integration user interfaces

- **Save new model**

  Operation ID: **com.teamcenter.bhm.model.save**

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
NAVIGATE_HOME_FOLDER	Disables navigation from the Home folder.	NA	NA	Home folder navigation is allowed.
NAVIGATE_CLASSIFICATION_TREE	Disables navigation from the classification tree.	NA	NA	Classification tree navigation is allowed.
SEARCH_TEAMCENTER	Disables Teamcenter search.	NA	NA	Teamcenter search is allowed.
SHOW_PRODUCT_STRUCTURE	Hides the product structure panel and disables the **Show structure panel** button.	Makes the Product structure panel visible. Disables the **Show Structure panel** button.	Makes the product structure panel visible, but users have the option of hiding it using the **Show Structure panel** button.	The Product structure panel is hidden, but users can view it using the **Show Structure panel** button.
SHOW_PROPERTIES	Hides the Properties panel and disables the **Show properties panel** button.	Shows the Properties panel. The **Show property panel** button is disabled.	Shows the Properties panel, but users can hide it using the **Show property panel** button.	The Properties panel is hidden, but users can view it using the **Show property panel** button.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
ENTER_ITEM_INFO	Disables manual entry or auto assigning of item ID, revision ID, and name.	NA	NA	Manual entry and auto assigning of item ID, revision ID, and name is enabled.
ADDITIONAL_DATA	Disables the ability to assign files or folders to model objects as additional data.	NA	NA	Users can assign files and folders to model data as additional data.
ADDITIONAL_DATA_FILE	Disables the ability to assign files to model data as additional data.	Only files can be assigned to model data as additional data. The ability to add a folder to models as additional data is disabled.	Enables the ability to add files to models as additional data.	Files and folders can be assigned to models as additional data.

- **Overwrite existing model**

  Operation ID: **com.teamcenter.bhm.model.overwrite**

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
ENTER_ITEM_INFO	Disables manual and auto assignment of item ID, revision ID, and name.	NA	NA	Manual and auto assignment of item ID, revision ID, and name is enabled.
ADDITIONAL_DATA	Disables the ability to assign files or folders to model objects as additional data.	NA	NA	Users can assign files and folders to model data as additional data.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
ADDITIONAL_DATA_FILE	Disables the ability to assign files to model data as additional data.	Only files can be assigned to model data as additional data. The ability to add folders to models as additional data is disabled.	Enables the ability to add files to models as additional data.	Files and folders can be assigned to models as additional data.
ADDITIONAL_DATA_REMOVE	Disables the ability to remove files associated with the model data as additional data.	NA	NA	Files associated with the model data as additional data can be removed.
ADDITIONAL_DATA_OVERWRITE	Disables the ability to overwrite files associated with the model data as additional data.	NA	NA	Files associated with the model data as additional data can be overwritten.
CHECK_IN_MODEL	Disables the check-in operation during a save.	Models and submodels are always checked in during the save operation.	The check box to check in the model is selected, but users can clear the check box.	The check box to check in the model is not selected. If users want to check in the model, they must select the check box.

- **Open model**

  Operation ID: **com.teamcenter.bhm.open**

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
NAVIGATE_HOME_FOLDER	Disables the navigation from the Home folder.	NA	NA	Home folder navigation is allowed.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
NAVIGATE_CLASSIFICATION_TREE	Disables the navigation from the classification tree.	NA	NA	Classification tree navigation is allowed.
SEARCH_TEAMCENTER	Disables Teamcenter search.	NA	NA	Teamcenter search is allowed.
SHOW_PRODUCT_STRUCTURE	Hides Product structure panel and disables the **Show Structure panel** button.	Makes the product structure panel visible. Disables the **Show Structure panel** button.	Makes the product structure panel visible but users have the option of hiding it using the **Show Structure panel** button.	Product structure panel is hidden but users can view it using the **Show Structure panel** button.
SHOW_PROPERTIES	Hides the Properties panel and disables the **Show property panel** button.	Shows the Properties panel. The **Show property panel** button is disabled.	Shows the Properties panel but users can hide the Properties panel using the **Show property panel** button.	The Properties panel is hidden but users can view it using the **Show property panel** button.
CHECKOUT_TOP_MODEL	Disables the ability to check out the top model and all sub-models.	Always checks out the top model but not the submodels.	The **Checkout Top Model** check box is selected. This allows users to clear it.	The **Checkout Top Model** check box is not selected. This allows users to select it.
CHECKOUT_TOP_MODEL_AND_ ALL_SUBMODELS	Disables the ability to check out the top model and all submodels.	Always checks out the top model and all submodels.	The **Checkout Top Model and All Submodels** check box is selected. This allows users to clear it.	The **Checkout Top Model and All Submodels** check box is not selected. This allows users to select it.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
DOWNLOAD_ADDITIONAL_DATA	Disables the ability to download additional data to the staging directory when opening the model from Teamcenter.	Always downloads additional data to the staging directory when opening the model from Teamcenter.	The **Download additional data** check box is selected.	The **Download additional data** check box is not selected.

- **Insert model**

  Operation ID for inserting a model within a model: **com.teamcenter.bhm.model.insertmodel**

  Operation ID for inserting an existing model in a subsystem: **com.teamcenter.bhm.subsystem.insertmodel**

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
NAVIGATE_HOME_FOLDER	Disables navigation from the Home folder.	NA	NA	Home folder navigation allowed.
NAVIGATE_CLASSIFICATION_TREE	Disables navigation from the classification tree.	NA	NA	Classification tree navigation allowed.
SEARCH_TEAMCENTER	Disables Teamcenter search.	NA	NA	Teamcenter search allowed.
SHOW_PRODUCT_STRUCTURE	Hides the Product structure panel and disables the **Show Structure panel** button.	Makes the Product structure panel visible. Disable the **Show Structure panel** button.	Makes the Product structure panel visible but users have the option of hiding it using the **Show Structure panel** button.	The Product structure panel is hidden but users can view it using the **Show Structure panel** button.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
SHOW_PROPERTIES	Hides the Properties panel and disables the **Show property panel** button.	Shows the Properties panel. The **Show property panel** button is disabled.	Shows the Properties panel but users can hide it using the **Show property panel** button.	The Properties panel is hidden but users can view it using the **Show property panel** button.
DOWNLOAD_ADDITIONAL_DATA	Disables the ability to download additional data to the staging directory when opening the model from Teamcenter.	Always downloads additional data to the staging directory when opening the model from Teamcenter.	The **Download additional data** check box is selected.	The **Download additional data** check box is not selected.

- **Checkout model**

  Operation ID: **com.teamcenter.bhm.model.checkout**

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
CHECKOUT_ALL_SUBMODELS	Disables the ability to check out all submodels.	Always checks out the submodels when checking out the top model.	The **Checkout All Submodels** check box is selected. This allows users to clear it.	The **Checkout All Submodels** check box is not not selected. This allows users to select it.
BACKUP_CURRENT_MODEL	Disables the ability to back up model data while checking out the model from Teamcenter.	Always backs up model data while checking out the model from Teamcenter.	Model data is backed up while checking out the model from Teamcenter but users have the option to not back up the data.	Model data is not backed up while checking out the model from Teamcenter but users have the option to back up the data.

- **Open from rich client**

  You need not update the integration definition file for overriding actions initiated from the rich client. To override actions:

  o Create a preference named **BHM_OPEN_FROM_RAC_UI_OVRD**.

  o Specify the user actions that must be overriden as the value of the preference, for example, **CHECKOUT_TOP_MODEL=NEVER**.

Preference name	NEVER	ALWAYS	DEFAULT	Default behavior
CHECKOUT_TOP_MODEL	Disables ability to check out the top model and all the submodels.	Always checks out the top model but not the submodels.	The **Checkout Top Model** check box is selected, allowing the users to clear it.	The **Checkout Top Model** check box is not selected, allowing the users to select it.
CHECKOUT_TOP_MODEL_AND_ALL_SUBMODELS	Disables ability to check out the top model and all submodels.	Always checks out the top model and submodels.	The **Checkout Top Model and All Submodels** check box is selected, the allowing users to clear it.	The **Checkout Top Model and All Submodels** check box is not selected, allowing the users to select it.
DOWNLOAD_ADDITIONAL_DATA	Disables the ability to download additional data to the staging directory when opening the model from Teamcenter.	Always downloads additional data to the staging directory when opening the model from Teamcenter.	The **Download additional data** check box is selected.	The **Download additional data** check box is not selected.

Configure the display of integration menu commands

You can specify which integration menu commands appear in the Teamcenter menu of MATLAB Simulink by specifying the menu configuration in a MATLAB script file. This script file contains a list of integration operation identifiers. These identifiers correspond to each integration operation.

The following is the list of the available integration operations and their corresponding integration operation identifiers:

Integration operation	Integration operation identifiers
Save model	**com.teamcenter.bhm.model.save**
Save library	**com.teamcenter.bhm.library.save**
Save As model	**com.teamcenter.bhm.model.saveas**
Save As library	**com.teamcenter.bhm.library.saveas**
Open library	**com.teamcenter.bhm.open**
Open library	**com.teamcenter.bhm.open**
Insert model in model	**com.teamcenter.bhm.model.insertmodel**
Insert model in library	**com.teamcenter.bhm.library.insertmodel**
Insert model in subsystem	**com.teamcenter.bhm.subsystem.insertmodel**
Insert subsystem block in model	**com.teamcenter.bhm.model.insertblock**
Insert subsystem block in library	**com.teamcenter.bhm.library.insertblock**
Insert subsystem block in subsystem	**com.teamcenter.bhm.subsystem.insertblock**
Revise model	**com.teamcenter.bhm.model.revise**
Revise library	**com.teamcenter.bhm.library.revise**
Traceability	**com.teamcenter.bhm.traceability**
Checkout model	**com.teamcenter.bhm.model.checkout**
Checkout library	**com.teamcenter.bhm.library.checkout**
Checkin model	**com.teamcenter.bhm.model.checkin**
Checkin library	**com.teamcenter.bhm.library.checkin**
Cancel checkout model	**com.teamcenter.bhm.model.cancelcheckout**
Cancel checkout library	**com.teamcenter.bhm.library.cancelcheckout**
Logoff	**com.teamcenter.bhm.logoff**

Configure the display of menu commands as follows:

1. Create a MATLAB m-script with the name **fn_getAvailableTeamcenterOperations**. In this m-script, add the integration operations that correspond to the Teamcenter menu commands.

Example

If you want the Teamcenter menu to only list the operations *save model, open, insert block, insert model, check-in, check-out,* and *cancel check-out*, update the script file as follows:

```
function availableOperationNameList =
fn_getAvailableTeamcenterOperations()
availableOperationNameList =
{
'com.teamcenter.bhm.model.save',...
'com.teamcenter.bhm.open',...
'com.teamcenter.bhm.model.insertmodel',...
'com.teamcenter.bhm.library.insertmodel',...
'com.teamcenter.bhm.subsystem.insertmodel',...
'com.teamcenter.bhm.model.insertblock',...
'com.teamcenter.bhm.library.insertblock',...
'com.teamcenter.bhm.subsystem.insertblock',...
'com.teamcenter.bhm.model.checkout',...
'com.teamcenter.bhm.library.checkout',...
'com.teamcenter.bhm.model.checkin',...
'com.teamcenter.bhm.library.checkin',...
'com.teamcenter.bhm.model.cancelcheckout',...
'com.teamcenter.bhm.library.cancelcheckout',...
'com.teamcenter.bhm.logoff'
};
return;
end
```

2. Deploy this m-script in the behavior modeling client at the following location:

   TC_ROOT/**bhm/matlab/scripts**

To verify the modified menu configuration for models, choose **Tools** →**Teamcenter** in the MATLAB Simulink model window, and confirm that only the menu operations that were specified in the m-script file are displayed.

To verify the modified menu configuration for a library, choose **Tools** →**Teamcenter** in the MATLAB Simulink library window, and confirm that only the menu operations that were specified in the m-script file are displayed.

Tip

To avoid tampering of the m-script file, convert the m-script to p-code before deploying.

Note

If the name of the m-script is wrong or if there are any errors in the m-script, you may view unexpected behavior, or Teamcenter menus might not be displayed.

Performing client-side configurations for MATLAB Simulink integration

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
```

Configuring integration settings

You need not configure these integration settings if you have a 4-tier installation.

- Modify the **BHMClient.properties** file to configure the behavior modeling tool integration settings for MATLAB Simulink. This file allows you to specify the Teamcenter server connection URL, identify the method of logging on, preserve a session through cookie files, and so on.

  This file is located in *TC_ROOT*/**bhm**.

- Modify the **start_matlab.bat** located in the *TC_ROOT*/**bhm/matlab/scripts** directory and **load.bat** located in *TC_ROOT*/**bhm/matlab** to specify the *TC_ROOT* environment variable, MATLAB ROOT, and the temporary directory.

- If you are deploying Teamcenter MBSE Integration Gateway version 4.1.1 onwards on Teamcenter version 12 and above and if your MATLAB version is below R2018b, create an environment variable called **MATLAB_JAVA** with a minimum value of JRE 1.8.

Note

When configuring the staging directory in the **BHMClient.properties** file, ensure that the directory name does not contain periods.

Add relative paths of models to the MATLAB Simulink search path

If the submodels and the top model are in different folders, add the relative paths of the submodel folders to the MATLALB search path. If you do not do this, you receive an error when opening or saving a model.

Add the relative folder paths of the submodel with respect to the top model in the MATLAB Simulink search path by using the **ModelProperties→Callbacks→PreLoadFcn** callback for the model.

Example

addpath '.\submodel1'

addpath '.\submodel2'

Note

If the folder path added to the MATLAB Simulink search path is incorrect, the data saved to Teamcenter is also be incorrect.

Allocate virtual memory for working with a large number of MATLAB models

While opening a MATLAB Simulink file from a folder that contains a large number of models, MATLAB Simulink may display an out-of-memory error. To avoid this, set the virtual memory being allocated to a higher level in the **java.opts** file. For example, use the **-Xmx1024M** value to allocate 1 GB of RAM.

If the **java.opts** file does not exist, create one and save it at a location of your choice, for example, *MATLAB_ROOT*/**bin/winx64**.

Caution

When working with the integration, if you set the **Xmx** value to a value higher than your system's RAM, you may see the following message:

```
Could not create the Java virtual machine.
Set the Xmx value to a lower setting that your system supports.
```

Verify the integration with MATLAB Simulink

Start MATLAB Simulink in *integration* mode by using the **start_matlab.bat** file located in the *TC_ROOT***bhm\matlab\scripts** directory.

Verify if you can perform model management operations such as opening a model or saving a model and so on.

Chapter 5: Configuring concurrent modeling

Performing server-side configurations for concurrent modeling

Configure what data is 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>`

Configure additional branch properties to import

When you import models from your computer to Teamcenter, you can import additional branch properties. On successful import, the branch properties are associated with the Teamcenter branch. 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 branch properties:

1. Create a property file that contains the properties 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 an updated the **name** argument with a value (**Description**). The value of the **name** argument must match the property in the property file that we created in the previous step.

   The property, in this example, 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. Run the import command with an argument that specifies the property file.

Example

```
import "D:\SimData\Powerwindow" PowerWindowBr -properties_file=
"C:\properties_input.txt"
```

The properties defined in the property file are imported and associated with the Teamcenter property on the branch object. Check the branch console to ensure that all the properties are imported.

Configure branch name uniqueness

To control the *uniqueness* of branch names, set the **BRANCH_BranchNameUniqueness_scope** preference to one of the following values:

- **RootBranch**

  (Default) Teamcenter ensures that a branch name is unique for each SysDM object or project.

- **Database**

  Teamcenter ensures that a branch name is unique in the entire database, for every SysDM object or project.

- **NoCheck**

  Teamcenter does not check whether a branch name is unique. Branches with the same name are permitted within an object, project, or the entire database.

This preference is applied when you create a new branch, copy an existing branch to a new branch, or modify the name of an existing branch. Teamcenter displays an error message if the configured uniqueness is violated.

Configure what object types are allowed in a branch

- To configure what object types are allowed in a branch, update the **BHV0_BranchContent_AllowedTypes** preference with information about the allowed object types, for example, **ItemRevision**.

- To specify the object types you want to exclude from a branch, add their names in braces next to the allowed object types, for example, **ItemRevision{CustomRevisionType1, CustomRevisionType2}**

- To exclude file types of a specific dataset type, specify the dataset name and the dataset type separated by two colons and the file types to be excluded in braces, for example, **Dataset::Misc{jpg, png}**

Example

Preference value	Setting
ItemRevision	Allows objects of the type ItemRevision and its children to be added to the branch.
ItemRevision{CustomRevisionType1, CustomRevisionType2}	Allow all objects of the type ItemRevision and its children to the branch except the objects CustomRevisionType1 and CustomRevisionType2.
Dataset{Text, jpg}	Allows all objects of the type *Dataset* and its children to the branch except the file types *Text* and *JPEG*.
Dataset::Misc{jpg, png}	Allows all objects of the type *Dataset* and its children to the branch except the file types *Text* and *JPEG* of the Dataset type Misc.

Performing client-side configurations for concurrent modeling

Enable the branch console by updating registry entries

Add the following entries to the Windows registry to enable branch console:

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/fnd0branch]
"Extension"=".bnv"

[HKEY_CLASSES_ROOT\.bnv]
"Content Type"="application/fnd0branch"
@="fnd0branchfile"

[HKEY_CLASSES_ROOT\fnd0branchfile]

[HKEY_CLASSES_ROOT\fnd0branchfile\shell]

[HKEY_CLASSES_ROOT\fnd0branchfile\shell\open]
@="&open"

[HKEY_CLASSES_ROOT\fnd0branchfile\shell\open\command]
@="<TC_ROOT>\\bhm\\openInConsole.bat \"%1\""
```

Add the following entries to the Windows registry to enable the merge functionality:

```
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/fnd0branchMerge]
"Extension"=".mrg"

[HKEY_CLASSES_ROOT\.mrg]
"Content Type"="application/fnd0branchMerge"
@="fnd0branchfileMerge"
```

```
[HKEY_CLASSES_ROOT\fnd0branchfileMerge]

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell]

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell\open]
@="&open"

[HKEY_CLASSES_ROOT\fnd0branchfileMerge\shell\open\command]
@="<TC_ROOT>\\bhm\\mergeBranchNodes.bat \"%1\""
```

Ensure that the staging directory is created

If you have not created the staging directory, ensure that you create the staging directory. After you create the staging directory, specify it's location in the **BHMClient.properties** file. This file is located in the *TC_ROOT***bhm** directory.

Warning

If you do not specify a staging directory, the interactions with the file system such as import, download, save will not function.

Chapter 6: Customizing Teamcenter MBSE Integration Gateway operations

Overview of customizing Teamcenter MBSE Integration Gateway operations

Teamcenter provides Java APIs and extensions to help you customize behavior modeling operations.

To use these APIs and extensions, you must install the following components:

- Behavior modeling common client
- Behavior modeling tool connector, for example, MATLAB connector.

For more information about installing these components, see *Overview of installing Teamcenter MBSE Integration Gateway*.

Customizing Teamcenter MBSE Integration Gateway operations by using APIs

Overview of customizing Teamcenter MBSE Integration Gateway operations by using APIs

Teamcenter provides Java APIs for the following behavior modeling operations:

- Open
- Pre-save
- Save
- Export
- Login
- Logout
- Derby cache status
- FCC status

You can invoke these APIs from any Java program or from the behavior modeling tool-specific script.

To use the APIs:

- Implement an external Java program or tool-specific script using the published APIs.

- Add all the .jar filesin the *Teamcenter_ROOT***bhm** directory to the Java build path libraries.
- If you are using MATLAB, ensure that you add the following entries to the PATH variable:
  - *MATLAB_ROOT***bin\win64**
  - *MATLAB_ROOT***sys\jxbrowser\win64\lib**
- Add **xercesImpl.jar** to the Java build path libraries.
- The class **OperationsAPIs** in the **com.teamcenter.behaviormodeling.commonclient.operations.api** package contains the behavior modeling integration APIs.
- Before using the APIs, you must establish a Teamcenter session using the login API.
- Use the utility functions to generate the xml input strings for the Save and the Open APIs. This utility is available in the **BHMOperationData** and the **BHMDesign** classes located in the **com.teamcenter.behaviormodeling.common.xml** package.

  You can use the Pre-save API to generate input files for the Save API. The xml file that the Pre-save API generates does not contain information about additional data. To save additional data, you must explicitly update the xml file.

Example

The Save API requires an XML string as input. This string contains information about the models to be saved. This XML string can be based on the **BHMOperationsSchema.xsd** file as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations xmlns="http://www.plmxml.org/Schemas/bhm">
 <operationData modelPath="D:\HSNModels">
  <modelInfo>
   <Model createdDate="Mon Sep 02 12:35:30 2013"
    tcType="Bhm0BehaviorModlRevision"
    instanceType="ModelReference" tcItemName="SingleModel"
    tcrevId="H"
    tcitemId="000205" tcuid="" modelIdentifier="SingleModel"
    modelName="SingleModel">
    <Attributes>
     <Key name="object_desc">
      <Value>My Desc</Value>
     </Key>
     <RevisionAttributes />
    </Attributes>
   </Model>
  </modelInfo>
  <options>
   <saveOperationOptions folderUID="g9UJCdVLopK8AC"
 checkIn="false" addAdditionalData="false" />
  </options>
 </operationData>
</BHMOperations
```

Operations API

This API is an overloaded constructor that accepts a Teamcenter connection object. The Teamcenter connection object passed through this constructor is used for all Behavior Modeling operations. If you are using this API, you need not use the bhmLogin API.

```
public OperationsAPIs (Connection connection, String serverURL, String toolType)
```

Inputs

connection
Specifies the Teamcenter connection object.

serverURL
Specifies the URL to the Teamcenter server.

toolType
Specifies the behavior modeling tool, for example, MATLAB.

bhmLogin API

You must start a Teamcenter session, using the bhmLogin API before using the Open and Save APIs.

```
BHMOperationOutput bhmLogin (String userName, String password,
 String group, String toolType)
```

Inputs

username
Specifies the Teamcenter user ID.

password
Specifies the password of the Teamcenter user.

group
(Optional) Specifies the group associated with the user.

toolType
Specifies the behavior modeling tool, for example, MATLAB.

Example

```
BHMOperationOutput bhmLogin ("infodba", "infodba", "", "MATLAB");
```

bhmOpenOperation API

Opens behavior models from Teamcenter.

```
List<BHMOperationOutput> bhmOpenOperation(List<String> dataOpenInputXML,
 String toolType)
```

Inputs

dataOpenInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string, using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelName** and **rootModelIdentifier**.
- In the **Model** element, specify the values for **modelName**, **modelIdentifier**, and **tcuid**.
- In the **openOperationOptions** element, specify the values of the following variables:
  - **checkout**

    To check out the root model, set the value to **true**.
  - **checkOutAll**

    To check out all submodels of the root model, set the value to **true**.
  - **addAdditionalData**

    To download additional data, set the value to **true**.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataOpenInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to open.

BHMOperationOutput variables	Description
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

Pass the following string to the **dataOpenInputXML** argument of the bhmOpenOperation API to load and check out a model.

To load multiple models, you must pass separate input XMLs to **dataOpenInputXML**.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations rootModelName="test" rootModelIdentifier="test"
xmlns="http://www.plmxml.org/Schemas/bhm">
  <modelInfo>
    <Model modelName="test" modelIdentifier="test" tcuid="Q2TJlnST4BflXC" />
  </modelInfo>
  <options>
    <openOperationOptions checkOut="true" checkOutAll="false"
     addAdditionalData="false" />
    </options></BHMOperations>
```

bhmPreSaveOperation API

This API generates an XML string that can be used as the input XML string for the Save API.

```
List<BHMOperationOutput> bhmPreSaveOperation(List<String>
 lstRootModelFullFilePath, String toolType)
```

Inputs

lstRootModelFullFilePath

Specifies the full path of the root model.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **lstRootModelFullFilePath**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**

BHMOperationOutput variables	Description
message	Specifies the cause of the API failure.
inputXML	Is not used.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd** that is used as input for the bhmSaveOperation API.
rootModelName	Specifies the root model.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

To generate XML files for two root model files, you can define the paths as follows:

```
List<String> paths = new ArrayList<String>();
   paths.add("D:\\matlabModels\\myModel11.slx");
   paths.add("D:\\matlabModels\\myModel21.slx");
```

bhmSaveOperation API

Use the following API to save and check in models to Teamcenter.

```
List<BHMOperationOutput> bhmSaveOperation (List<String> dataSaveInputXML,
 String toolType)
```

Inputs

dataSaveInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

Do not specify files that do not exist or are internal data such as the ModelFolder organization file. If you specify such files, you may get an error.

You can generate the xml string using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelFolder**, **rootModelName**, and **rootModelIdentifier**.
- In the **Model** element:
  - Specify the values for **modelName** and **modelIdentifier**.
  - Specify the value of **tcuid** for an existing model.

    Do not specify a value for **tcuid** in new models.

- In the **Attributes** element, specify the value of the **item_id** for items and revisions. If you do not specify the IDs, Teamcenter automatically generates them.

  Do not specify values for **tcrevId** and **tcitemId** as they are for internal use.

```
<Attributes>
  <Key name="item_id">
    <Value>001001</Value>
  </Key>
  <RevisionAttributes>
    <Key name=" item_id ">
      <Value>P</Value>
    </Key>
  </RevisionAttributes>
</Attributes>
```

  You can also specify the description of the model in the **Attributes** element as follows:

```
<Attributes>
  <Key name="object_desc">
    <Value>This is S1 Desc</Value>
  </Key>
</Attributes>
```

- Specify the submodel information in the **ModelElement** element.

  Specify the ID and revision of the submodel in the **Attributes** element of **ModelElement**.

  If you do not specify the ID and revision for the submodel, Teamcenter automatically generates them.

- In the **saveOperationOptions** element, specify the values of the following variables:

  - **folderUID**

    Saves the model to the Teamcenter folder specified by the folder UID.

  - **checkIn**

    To check in the root model, set the value to **true**.

  - **addAdditionalData**

    To attach additional data, set the value to **true**, and specify the additional data details in the **AdditionalData** element.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataSaveInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to be saved.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

To save a new model to Teamcenter, you can pass the following string through the **dataSaveInputXML** list of the bhmSaveOperation API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations xmlns="http://www.plmxml.org/Schemas/bhm"
  rootModelFolder="D:\Models\MatlabModels" rootModelName="MyModel"
  rootModelIdentifier="MyModel">
  <modelInfo>
    <Model toolId="" teamcenterVersion="1.1" tcType="Bhm0BehaviorModlRevision"
      tcItemName="MyModel" tcrevId="" tcitemId="" tcuid=""
      modelIdentifier="MyModel"
      modelName="MyModel">
      <Attributes>
        <Key name="object_name">
          <Value>MyModel</Value>
        </Key>
        <Key name="object_desc">
          <Value></Value>
        </Key>
        <Key name="item_id">
          <Value>000041</Value>
        </Key>
        <RevisionAttributes>
          <Key name="item_revision_id">
            <Value>A</Value>
          </Key>
        </RevisionAttributes>
      </Attributes>
      <AdditionalData>
        <File description="" relationType="Bhm0AdditionalData"
          reservation="true" path="D:\Models\MatlabModels\AdditionalData"
          fileext="txt" fileName="ReviewComments.txt" namedReferenceType="Text"
          tcDatasetType="Text" tcuid="" action="add" />
      </AdditionalData>
```

```
      <Contents>
        <ModelElements>
          <ModelElement parent="S1" tcType="Bhm0BehaviorModlRevision"
            instanceType="MATLAB:ModelReference" instanceUid="" tcuid=""
            modelIdentifier="MyChild" modelName="MyChild" name="Model">
            <Attributes>
              <Key name="object_name">
                <Value>MyChild</Value>
              </Key>
              <Key name="object_desc">
                <Value>This is MyChild Desc</Value>
              </Key>
              <Key name="item_id">
                <Value>001003</Value>
              </Key>
              <RevisionAttributes>
                <Key name="item_revision_id">
                  <Value>Q</Value>
                </Key>
              </RevisionAttributes>
            </Attributes>
          </ModelElement>
        </ModelElements>
      </Contents>
    </Model>
  </modelInfo>
  <options>
    <saveOperationOptions folderUID="" checkIn="true"
      addAdditionalData="true" />
  </options>
</BHMOperations>
```

bhmSaveAsOperation API

Use the following API to create a copy of an existing model and save it to Teamcenter.

If the model has reusable blocks, the SaveAs operation creates new copies of the reusable blocks.

```
List<BHMOperationOutput> bhmSaveAsOperation (List<String> dataSaveAsInputXMLs,
 String toolType)
```

Inputs

dataSaveInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

Note

You can also generate the input XML string, using the bhmPreSaveOperation API.

Provide the details of the model to be copied, such as the staging directory, and generate the XML file and update the output of this API.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelFolder**, **rootModelName**, and **rootModelIdentifier**.
- In the **Model** element:
  - Specify the values for **modelName** and **modelIdentifier**.
  - Specify the value of **tcuid**.
- In the **Attributes** element, specify the value of **item_id**, **item_revision_id**, and **object_name**. If you do not specify these IDs, Teamcenter automatically generates them.

```
<Attributes>
  <Key name="item_id">
    <Value>000300</Value>
  </Key>
  <RevisionAttributes>
    <Key name="object_name">
      <Value>M11SaveAs</Value>
    </Key>
    <Key name="item_revision_id">
      <Value>D</Value>
    </Key>
  </RevisionAttributes>
</Attributes>
```

- In the **saveOperationOptions** element, specify values for the following variables:
  - **folderUID**

    Saves the model to the Teamcenter folder specified by the folder UID.
  - **checkIn**

    Set the value to **false** as you cannot check in a new model.
  - **addAdditionalData**

    To copy additional data from the original model to the new model, set the value to **true**.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataSaveInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to be saved.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

To create a copy of a model and save it to Teamcenter, you can pass the following string through the **dataSaveInputXML** list of the bhmSaveOperation API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations rootModelFolder="D:\staging_11\M11_000260_A\M112" rootModelName="M11"
 rootModelIdentifier="M11"xmlns="http://www.plmxml.org/Schemas/bhm">
  <modelInfo applicationName="MATLAB">
    <Model modelName="M11" modelIdentifier="M11" tcuid="g_Zhgh35IvwBJA"
    tcitemId="000295" tcrevId="A" tcItemName="M11" instanceType="ModelReference"
    tcType="Bhm0BehaviorModl" lastSavedTimeStamp="0"
    createdDate="Tue May 12 12:24:22 2015" isRoot="true"
    containerModel="true" objectType="Model">
   <Attributes>
  <Key name="item_id">
   <Value>000300</Value>
  </Key>
  <RevisionAttributes>
   <Key name="object_name">
    <Value>M11SaveAs</Value>
   </Key>
   <Key name="item_revision_id">
    <Value>D</Value>
   </Key>
  </RevisionAttributes>
```

```
    </Attributes>
    <Contents>
   <ModelElements/>
    </Contents>
  </Model>
   </modelInfo>
   <options>
     <saveOperationOptions folderUID="AWSdOLciIvwBJA" checkIn="false"
     addAdditionalData="true"/>
   </options>
</BHMOperations>
```

bhmOpenBlockLibrary API

Opens behavior library models from Teamcenter.

```
List<BHMOperationOutput> bhmOpenBlockLibrary(List<String> dataOpenInputXML,
 String toolType)
```

Inputs

dataOpenInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string, using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelName** and **rootModelIdentifier**.
- In the **Model** element, specify the values for **modelName**, **modelIdentifier**, and **tcuid**.
- In the **openOperationOptions** element, specify the values of the following variables:
  - **checkout**

    To check out the root model, set the value to **true**.
  - **checkOutAll**

    To check out all submodels of the root model, set the value to **true**.
  - **addAdditionalData**

    To download additional data, set the value to **true**.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataOpenInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to open.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

Pass the following string to the **dataOpenInputXML** argument of the bhmOpenOperation API to load and check out a library model.

To load multiple library models, you must pass separate input XMLs to **dataOpenInputXML**.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations rootModelName="test" rootModelIdentifier="test"
xmlns="http://www.plmxml.org/Schemas/bhm">
  <modelInfo>
    <Model modelName="test" modelIdentifier="test" tcuid="Q2TJlnST4BflXC" />
  </modelInfo>
  <options>
    <openOperationOptions checkOut="true" checkOutAll="false"
     addAdditionalData="false" />
    </options></BHMOperations>
```

bhmSaveBlockLibrary API

Use the following API to save and check in library models to Teamcenter.

```
List<BHMOperationOutput> bhmSaveBlockLibrary (List<String> dataSaveInputXML,
 String toolType)
```

Inputs

dataSaveInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelFolder**, **rootModelName**, and **rootModelIdentifier**.
- In the **Model** element:
  - Specify the values for **modelName** and **modelIdentifier**.
  - Specify the value of **tcuid** for an existing model.

    Do not specify the value of **tcuid** for new models.
- In the **Attributes** element, specify the value of **item_id** for items and revisions. If you do not specify values for these, Teamcenter automatically generates them.

  If you do not specify values for **tcrevId** and **tcitemId** as they are for internal use.

```
<Attributes>
  <Key name="item_id">
    <Value>001001</Value>
  </Key>
  <RevisionAttributes>
    <Key name=" item_id ">
      <Value>P</Value>
    </Key>
  </RevisionAttributes>
</Attributes>
```

  You can also specify a description for the model in the **Attributes** element as follows:

```
<Attributes>
  <Key name="object_desc">
    <Value>This is S1 Desc</Value>
  </Key>
</Attributes>
```

- Specify the submodel information in the **ModelElement** element.

  Specify the ID and revision of the submodel in the **Attributes** element of **ModelElement**.

  You need not specify the ID and revision for the submodel, Teamcenter automatically generates them.
- In the **saveOperationOptions** element, specify the values of the following variables:
  - **folderUID**

    Saves the model to the Teamcenter folder specified by the folder UID.
  - **checkIn**

Checks in the root model when the value is set to **true**.

- **addAdditionalData**

  Attaches additional data when the the value is set to **true**, and specify the additional data details in the **AdditionalData** element.

Outputs

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataSaveInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to be saved.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the complete path of the root model.

Example

To save a new library model to Teamcenter, you can pass the following string through the **dataSaveInputXML** list of the bhmSaveBlockLibrary API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations xmlns="http://www.plmxml.org/Schemas/bhm"
  rootModelFolder="D:\Models\MatlabModels" rootModelName="MyModel"
  rootModelIdentifier="MyModel">
  <modelInfo>
    <Model toolId="" teamcenterVersion="1.1" tcType="Bhm0BehaviorModlRevision"
      tcItemName="MyModel" tcrevId="" tcitemId="" tcuid=""
      modelIdentifier="MyModel"
      modelName="MyModel">
      <Attributes>
        <Key name="object_name">
          <Value>MyModel</Value>
        </Key>
        <Key name="object_desc">
          <Value></Value>
```

```
          </Key>
          <Key name="item_id">
            <Value>000041</Value>
          </Key>
          <RevisionAttributes>
            <Key name="item_revision_id">
              <Value>A</Value>
            </Key>
          </RevisionAttributes>
        </Attributes>
        <AdditionalData>
          <File description="" relationType="Bhm0AdditionalData"
            reservation="true" path="D:\Models\MatlabModels\AdditionalData"
            fileext="txt" fileName="ReviewComments.txt" namedReferenceType="Text"
            tcDatasetType="Text" tcuid="" action="add" />
        </AdditionalData>
        <Contents>
          <ModelElements>
            <ModelElement parent="S1" tcType="Bhm0BehaviorModlRevision"
              instanceType="MATLAB:ModelReference" instanceUid="" tcuid=""
              modelIdentifier="MyChild" modelName="MyChild" name="Model">
              <Attributes>
                <Key name="object_name">
                  <Value>MyChild</Value>
                </Key>
                <Key name="object_desc">
                  <Value>This is MyChild Desc</Value>
                </Key>
                <Key name="item_id">
                  <Value>001003</Value>
                </Key>
                <RevisionAttributes>
                  <Key name="item_revision_id">
                    <Value>Q</Value>
                  </Key>
                </RevisionAttributes>
              </Attributes>
            </ModelElement>
          </ModelElements>
        </Contents>
      </Model>
    </modelInfo>
    <options>
      <saveOperationOptions folderUID="" checkIn="true"
        addAdditionalData="true" />
    </options>
  </BHMOperations>
```

bhmExportOperation API

Downloads behavior models, related datasets, additional data, and referenced models from Teamcenter to the specified folder. This API also restores the original folder structure and unzips the additional data that was zipped during the save operation.

```
List<BHMOperationOutput> bhmExportOperation(List<String> dataExportInputXMLs)
```

Input

dataExportInputXMLs

Specifies a list of xml input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string, using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the path of the folder where the model is to be exported in the **rootModelFolder** variable.
- In the **modelInfo** element, specify values for the following variables: **applicationName** and **tcuid**.
  - **applicationName**

    Specifies the name of the behavior modeling application, for example, **MATLAB**.
  - **tcuid**

    Specifies the Teamcenter ID of the model.
- In the **exportOperationOptions** element, specify the values of the following variable:
  - **getAdditionalData**

    To download additional data, set the value to **true**.

Output

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataOpenInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to export.
rootModelIdentifier	Specifies the identifier of the root model to be exported.

Example

Pass the following string to the **dataExportInputXML** argument of the bhmExportOperation API to export a model.

To load multiple models, you must pass separate input XMLs to **dataOpenInputXML**.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations xmlns="http://www.plmxml.org/Schemas/bhm rootModelFolder="test">
  <modelInfo applicationName="MATLAB">
    <Model tcuid="Q2TJlnST4BflXC" />
  </modelInfo>
  <options>
    <openOperationOptions checkOut="true" checkOutAll="false"
     addAdditionalData="false" />
    </options></BHMOperations>
```

bhm0GetModelOrgData API

This is a server API that returns the model organization data that is saved in the model object in Teamcenter. The model organization data consists of a model folder name, relative paths of all the submodel files, and model-configured data. This API is available for the Bhm0BehaviorModelRevision and Bhm0CompRepositoryRevision objects and all inherited business objects.

The model organization information is returned for the model object on which the API is invoked. To obtain the organization information for all the submodels, the API should be invoked recursively on all the submodels.

```
int bhm0GetModelOrgData ( behaviormodeling::BhmModelOrgDataS *modelOrgData )
```

Inputs

BhmModelOrgDataS

Specifies a reference to an empty **BhmModelOrgDataS** structure object. The API returns model organization data in this input structure object.

Outputs

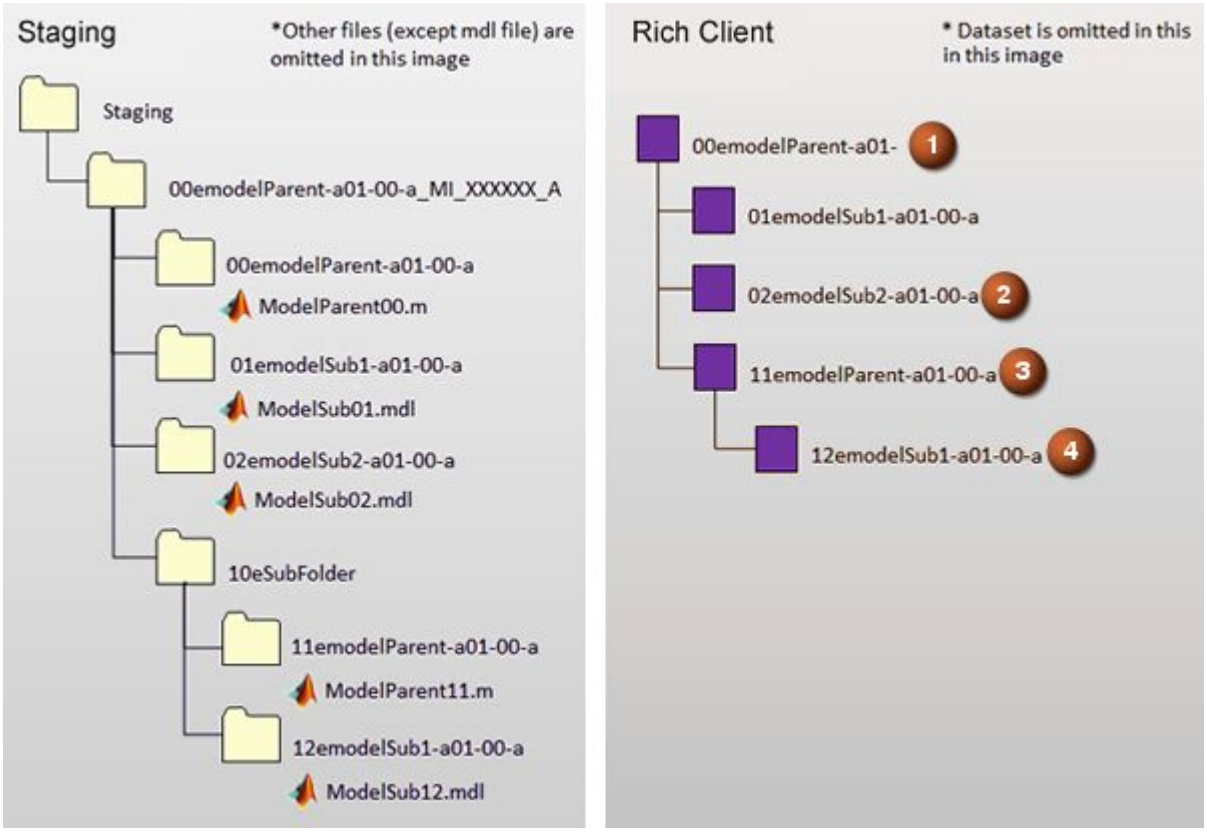
The API outputs the model organization data in the **BhmModelOrgDataS** structure object.

The **BhmModelOrgDataS** structure object returns the following:

Output object	Description
modelName	Specifies the name of the model.
modelFolderPath	Specifies the model path relative to the model folder.
configFilesFolderPath	Specifies the paths of the configuration data files relative to the model, including the name of the file.
mapSubModelFilesFolderPath	Specifies the submodel name and its path relative to the parent model exluding the submodel folder name.

Example

Assume that your model organization appears as follows in the staging directory and the rich client.



On using the API, you obtain the following output:

Output object	Object name
modelName	**ModelParent00**
modelFolderPath	**00emodelParent-a01-00-a**
configFilesFolderPath	**./ const / SomeDataModel.mat**
	./ simdata /SimDataSpec.doc
	./pic/SimResult.jpg
	./extern/ ExternDataModel.m
mapSubModelFilesFolderPath	**ModelSub01 - ../**
	ModelSub02 - ../
	ModelParent11 - ../10eSubFolder
modelName	**ModelSub02**
modelFolderPath	**02emodelSub2-a01-00-a**

Output object	Object name
configFilesFolderPath	**./ const / SomeDataModel.mat**
	./ simdata /SimDataSpec.doc
	./pic/SimResult.jpg
	./extern/ ExternDataModel.m
mapSubModelFilesFolderPath	
modelName	**ModelParent11**
modelFolderPath	**11emodelParent-a01-00-a**
configFilesFolderPath	**./ const / SomeDataModel.mat**
	./ simdata /SimDataSpec.doc
	./pic/SimResult.jpg
	./extern/ExternDataModel.m
mapSubModelFilesFolderPath	**ModelSub12 - ../**
modelName	**ModelSub12**
modelFolderPath	**12emodelSub1-a01-00-a**
configFilesFolderPath	**./ const / SomeDataModel.mat**
	./ simdata /SimDataSpec.doc
	./pic/SimResult.jpg
	./extern/ExternDataModel.m
mapSubModelFilesFolderPath	

bhmReviseOperation API

Use the following API to revise models that are managed in Teamcenter.

```
List<BHMOperationOutput> bhmReviseOperation (List<String> dataReviseInputXML,
 String toolType)
```

Inputs

dataReviseInputXML

Specifies a list of input strings that are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string, using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelName** and **rootModelIdentifier**.
- In the **Model** element, specify the values for **modelName**, **modelIdentifier**, and **tcuid**.
- In the **reviseOperationOptions** element, specify the values of the following variables:
  - **revisionID**

    Specifies the revision ID to be set when revising the behavior model. If this value is empty, the revision ID is automatically assigned. This value must contain alphanumeric characters and must not contain the _ (underscore) character or the . (period) character.

  - **carryForwardAdditionalData**

    Specifies whether additional data must be revised and carried forward to the revised model. The value of this variable is a boolean parameter. If the value if **true**, additional data is carried forward. If not, the revised model will not have any additional data.

Output

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataReviseInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
outputXML	Specifies an XML string based on **BHMDesignSchema.xsd**.
rootModelName	Specifies the root model to be saved.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.

Example

To revise a model, you can pass the following string through the **dataReviseInputXML** list of the **bhmReviseOperation** API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations rootModelName="test" rootModelIdentifier="test"
   xmlns="http://www.plmxml.org/Schemas/bhm">
  <modelInfo>
```

```
    <Model modelName="test" modelIdentifier="test"
     tcuid="Q2TJlnST4BflXC" />
  </modelInfo>
  <options>
    <reviseOperationOptions revisionID="D"
    carryForwardAdditionalData="false" />
  </options>
</BHMOperations>
```

bhmCancelCheckOutOperation API

Use the following API to cancel the checkout of a model saved in Teamcenter.

```
List<BHMOperationOutput> bhmCancelCheckOutOperation
(List<String> dataCancelCheckOutInputXML,  String toolType)
```

Inputs

dataCancelCheckOutInputXML

Specifies a list of input strings with information about the models whose checkout must be canceled. These strings are based on the **BHMOperationsSchema.xsd** file.

You can generate the xml string, using the JAXB utility available with the **BHMOperationData** and **BHMDesign** classes in the **com.teamcenter.behaviormodeling.common.xml** package.

toolType

Specifies the behavior modeling tool, for example, MATLAB.

In your Java code, you must specify the following:

- In the **BHMOperations** element, specify the values for **rootModelFolder** and **rootModelIdentifier**.
- In the **Model** element:
  - Specify the values for **modelName** and **modelIdentifier**.
  - Specify the value of **tcuid** for an existing model.
- In the **ReservationOperationOptions** element, specify the value for **BackUpExistingModel**. This value backs up the model before the checkout is canceled.

Output

The output is a list of **BHMOperationOutput** objects.

The **BHMOperationOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

Each **BHMOperationOutput** object in the list corresponds to the input XML strings sent through the input parameter **dataCancelCheckOutInputXML**.

BHMOperationOutput variables	Description
OperationStatus	Returns one of the following values: **OPERATION_FAILED** **OPERATION_WARNING** **OPERATION_SUCCESSFUL** **OPERATION_SUCCESSFUL_WITH_INFO**
message	Specifies the cause of the API failure.
inputXML	Specifies the input XML string based on **BHMOperationsSchema.xsd**.
rootModelName	Specifies the root model to be saved.
rootModelIdentifier	Specifies the identifier of the root model.
rootModelFullFilePath	Specifies the full path of the root model.
backUpDirectory	Specifies the directory to back up the model before canceling the checkout.

Example

To cancel a checkout, you can pass the following string through the **dataCancelCheckOutInputXML** list of the **bhmCancelCheckOutOperation** API.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations rootModelFolder="D:\staging_11\M11_000260_A\M112"
rootModelIdentifier="M11"
xmlns="http://www.plmxml.org/Schemas/bhm">
  <modelInfo applicationName="MATLAB">
    <Model modelName="M11" modelIdentifier="M11" tcuid="g_Zhgh35IvwBJA">
   <Attributes/>
   <Contents>
  <ModelElements/>
   </Contents>
 </Model>
  </modelInfo>
  <options>
    <reservationOperationOptions backUpExistingModel="true/>
  </options>
</BHMOperations>
```

bhmLogout API

Use this API to end the Teamcenter session from the behavior modeling integration.

```
BHMOperationOutput bhmLogout()
```

getStateOfCurrentModel API

Provides information about the model that is cached in the derby database.

When modeling tool artifacts are saved in Teamcenter, model dependencies and some of the metadata, such as attributes, are saved as properties of Teamcenter objects. When the model is downloaded to a staging directory, the modeling tool refers to only the files associated with the model

objects. When these models are saved back to Teamcenter, the information about which Teamcenter objects these model files are associated with is cached in a derby database on the client machine. The derby database also keeps track of the checkout status of the model object.

This API provides the model information (TcUID) and the checkout status and is available in the **com.teamcenter.behaviormodeling.commonclient.operations.api** package.

```
BHMObjectStatusOutput getStateOfCurrentModel( String modelIdentifier,
String modelFileFullPath, String strToolType)
```

Inputs

modelIdentifier

Specifies the model identifier.

modelFileFullPath

Specifies the full path of the model

strToolType

Specifies the name of the modeling tool, for example, MATLAB.

Outputs

The output is a list of **BHMObjectStatusOutput** objects.

The **BHMObjectStatusOutput** class is available in the **com.teamcenter.behaviormodeling.common** package.

The output object has the following information:

BHMOperationOutput variables	Description
String modelIdentifier	Specifies the model identifier.
String modelDirectory	Specifies the full path of the model.
String tcUID	Species the UID of the model.
String checkoutStatus	Specifies the checkout status. It can be one of the following: **MODEL_NOT_IN_TEAMCENTER** **NOT_CHECKEDOUT_BY_ANYONE** **CO_BY_DIFFERENT_USER** **CO_BY_SAME_USER_IN_TEAMCENTER_ONLY** **CO_BY_SAME_USER_IN_TEAMCENTER_AND_BHM_IN_DIFFERENT_WORKING_DIR** **CO_BY_SAME_USER_IN_TEAMCENTER_AND_BHM_IN_SAME_WORKING_DIR**
TCMEOperationOutput. OperationStatus operationStatus	This is a string message. The message status is either **SUCCESS** or **FAILURE**. In the case of **SUCCESS**, the string is empty, but in the case of **FAILURE**, the string contains the error message.

Example

```
BHMObjectStatusOutput getStateOfCurrentModel
( "BHM_Save002", ModellingGatewayDataStore.getCachedObjectsModel(rootModelName).
getCachedModel().getPrimaryData().getFile().get(0).getFilePath(),"MATLAB")
```

ModelManagementFCCAPIs API

Provides the status of the File Client Cache (FCC).

The FCC deals with uploading, downloading, and caching files saved in Teamcenter. For any integration operation to be successful it is important to understand the status of FCC.

This API allows you to initialize the FCC, stop the FCC, and get its status.

For this API to work ensure the following:

- The following entry is added to the PATH environment variable: `%FMS_HOME%\bin`
- The **FMS_HOME** environment variable is configured.

you must add the location of FMS HOME to the path as follows: `%FMS_HOME%\bin`

```
TCMEFCCOutput out = ModelManagementFCCAPIs.tcmeInitializeFCC();
TCMEFCCOutput out = ModelManagementFCCAPIs.tcmeStopFCC();
TCMEFCCOutput out = ModelManagementFCCAPIs.getFCCStatus();
```

Inputs

None.

Outputs

The output is a **TCMEFCCOutput** object. It contains the following information:

BHMOperationOutput variables	Description
Enum out.getOperationStatus	Specifies if the operation has failed or succeeded. Returns the following: • **OPERATION_FAILED** • **OPERATION_SUCCESSFUL**

BHMOperationOutput variables	Description
List out.getFccInfo	Provides additional information about the FCC status. For the **tcmeInitializeFCC** method: • **FCC Started** This indicates that the FCC is successfully started through the API. • **Assigned FSC FCC URL is currently active** For the **tcmeStopFCC** method: • **FCC Stopped** This indicates that the FCC is successfully stopped through API. • **FCC Clients are connected to the FCC segment cache. Stop all client applications or use the '-kill' option.** If FCC is not stopped. For the **getFCCStatus** method: • **FCC Started** This indicates that the assigned FSC FCC URL is currently active. • **FCC Offline** This indicates that the FCC is offline.
Boolean out.isFccOnline	Returns true if the FCC is online and false if it is offline.

Rich client APIs

BHMOperations API

```
BHMOperations (String strToolType, TCComponent modelrevision)
```

This API is an abstract base class for all the behavior modeling operations supported through the rich client. The constructor considers the model tool name and the model revision object for which the behavior modeling operations are to be performed.

Inputs

strToolType

Specifies the name of the modeling tool, for example, MATLAB.

modelrevision

Specifies a reference to a model revision or a BOMLine of the model revision.

BHMModelOrgOperations::bhmSetFolderName API

```
void setFolderName( String strFolderName ) throws TCException,
 BHMCommonException, JAXBException
```

Using this API, a custom environment can update the folder organization metadata for new files that are associated to a model and that are expected to be downloaded in the configured folders.

If the input folder name and the relation with which the dataset is associated to the model revision object do not match, an error is displayed. The API supports the model revision object or the BOMLine of a model revision object.

Inputs

strFolderName

Specifies the name of the folder.

BHMModelOrgOperations::bhmSetFileRelativePath API

```
void setFileRelativePath( TCComponent dataset, String configFolderName,
String relativePath )
 throws BHMCommonException, TCException, JAXBException
```

Using this API, a custom environment can update the file organization meta-data for new files that are associated to a model and that are expected to be downloaded in the configured folders.

If the input folder name and the relation with which the dataset is associated to the model revision object do not match, an error is displayed. The API supports the model revision object or the BOMLine of a model revision object.

Inputs

dataset

Specifies the dataset corresponding to the file.

configFolderName

Specifies the name of the configured folder. This folder name must match the folder name from the integration definition file.

relativePath

Specifies the relative location of the file with the relation to the model folder.

Concurrent modeling APIs

tcmeImportModelsToBranch API

Imports models and associated data from the user workstation to the specified branch in Teamcenter.

```
ProjectDataOperationStatus tcmeImportModelsToBranch(String branchName,
String projectRootDirectory)
```

Input

branchName

Specifies the name of the banch object to be created in Teamcenter.

projectRootDirectory

Specifies the context folder or location from which the models will be imported.

Output

ProjectDataOperationStatus

Indicates the status of the branch import operation.

Example

```
TCMEProjectManagementAPIs projectAPIs = new TCMEProjectManagementAPIs
("<username>"<password>","");
projectAPIs.tcmeImportModelsToBranch( "ProjectName",
"D:\\MyProjects\\ProjectFolder" );
```

tcmeExportModelsFromBranch API

Exports the contents of a specified branch from Teamcenter to the local file system.

```
ProjectDataOperationStatus tcmeExportModelsFromBranch(String branchUid,
String strExportDirectoryOfBranch)
```

Inputs

branchUid

Specifies the unique indentifier of the branch to be exported.

strExportDirectoryOfBranch

Specifies the location where the branch data must be exported. This is an optional argument. If this argument is not specified, the branch contents are exported to the staging directory.

Output

ProjectDataOperationStatus

Indicates the status of the branch import operation.

Example

```
TCMEProjectManagementAPIs projectAPIs = new TCMEProjectManagementAPIs
("<username>"<password>","");
projectAPIs.tcmeExportModelsFromBranch("<project/branchuid>,
<Download folder for project>);
```

tcmeCheckoutModelFromBranch API

Checks out the specified model from a branch in Teamcenterr.

```
ProjectDataOperationStatus tcmeCheckoutModelFromBranch(String branchUid,
List<String> lstFilePath)
```

Inputs

branchUid

Specifies the unique indentifier of the branch to be checked out.

lstFilePath

Specifies the names of an input folder or file to be checked out.

Output

ProjectDataOperationStatus

Indicates the status of the checkout operation.

Example

```
TCMEProjectManagementAPIs projectAPIs = new TCMEProjectManagementAPIs
("<username>"<password>","");
// List of files to checkout
List<String> lstFileToCheckout = new ArrayList<String>();
lstFileToCheckout.add("D:\\MyProjects\\ProjectFolder\\Matlab\\RootModel.slx ");
lstFileToCheckout.add("D:\\MyProjects\\ProjectFolder\\Amesim\\Amesim.bame ");
projectAPIs.tcmeCheckoutModelFromBranch("ADSpBf1io7walD", lstFileToCheckout);
```

tcmeUpdatetModels API

Updates specified models from a branch in Teamcenter. This API also checks in the model to Teamcenter.

```
ProjectDataOperationStatus tcmeUpdatetModels(List<String> lstModelFilePaths,
boolean isModelToBeCheckedIn)
```

Inputs

lstModelFilePaths

Specifies the list of model files to be updated.

isModelToBeCheckedIn

Specifies if the model is to be checked in.

Output

ProjectDataOperationStatus

Indicates the status of the update operation.

Example

```
TCMEProjectManagementAPIs projectAPIs = new TCMEProjectManagementAPIs
("<username>"<password>","");
// List of files to update
List<String> lstFileToUpdate= new ArrayList<String>();
lstFileToUpdate.add("D:\\MyProjects\\ProjectFolder\\Matlab\\RootModel.slx ");
lstFileToUpdate.add("D:\\MyProjects\\ProjectFolder\\Amesim\\Amesim.bame ");
// Chekin the models after updating
projectAPIs.tcmeUpdatetModels(lstFileToUpdate,true);
```

Analysis Request APIs

getAttributeData API

This API extracts the input and output attribute information for the specified analysis request or study object.

Inputs

InputRevisions

Specifies a vector of the analysis request revision UIDs, study revision UIDs, or a mix of both.

Output

operationStatus

Specifies the following:

- Status of the operation
- Response data corresponding to the input analysis request or study revision

setAnalysisRequestData API

This API updates the attribute values based on the simulation results in Teamcenter. It also uploads result data in the form of result files. These files are then associated with the analysis request or the study revision.

Inputs

InputAttributeData

Specifies a vector of attribute values. These values are generated after the simulation. The attribute values are grouped per analysis request or study revision. You can also specify SMC files to be associated with the analysis request of study objects.

Outputs

operationStatus

Specifies the following:

- Status of the operation
- Response data corresponding to the input analysis request or study revision

getAnalysisRequestData API

This API is a wrapper around the Teamcenter SOA that extracts the analysis request data. The API returns the entire analysis request data, including the input-output objects, input-output attributes, all study objects and their data, end objects that are assoicated through trace links to the system blocks that are added to the analysis request, analysis request system blocks hierarchy up to the root node, and the associated data such as requirements and test cases. The API also provides an option to filter the unwanted data.

Inputs

InputRevisions

Specifies a vector of analysis request revision UIDs, study revision UIDs, or a mix of both.

ARDataFilter

Specifies an argument to filter unwanted data.

Outputs

operationStatus

Specifies the following:

- Status of the operation
- Response data corresponding to the input analysis request or study revision

Simcenter System Synthesis APIs

createOrUpdateDictionary API

This API creates or updates a list of dictionary objects in Teamcenter.

Inputs

CreateOrUpdateDictionaryInputData

Specifies an array of the **CreateOrUpdateDictionaryInputData** class elements that contains information such as the dictionary name, description, and file path.

Outputs

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation

- Response data corresponding to the new dictionary object UIDs. The response data is based on the class **SystemSynthesisModelingResponseData**.

createOrUpdateTemplateContainer API

This API creates or updates a list of Teamcenter Library objects in Teamcenter.

Inputs

CreateOrUpdateTemplateLibraryInputData

Specifies an array of the **CreateOrUpdateTemplateLibraryInputData** class that contains information such as the file path, proxy objects, related secondary objects.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the object UIDs of the new template container. The response data is based on the class **CreateOrUpdateTemplateLibraryResponseData**.

createOrUpdateBaseArchitecture API

This API creates or updates a list of base architecture objects in Teamcenter.

Inputs

CreateOrUpdateBaseArchitectureInputData

Specifies an array of the **CreateOrUpdateBaseArchitectureInputData** class that contains information such as the file path, related secondary object UIDs.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the new base architecture object UIDs. The response data is based on the class **SystemSynthesisModelingResponseData**.

createOrUpdateModelContainer API

This API creates or updates a list of model container objects in Teamcenter.

Inputs

CreateOrUpdateModelContainerInputData

Specifies an array of the **CreateOrUpdateModelContainerInputData** class elements that contains information such as the file path, proxy objects, and secondary object UIDs.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the new container object UIDs. The response data is based on the class **CreateOrUpdateModelContainerResponseData**.

createOrUpdateProject API

This API creates or updates a list of project objects in Teamcenter.

Inputs

CreateOrUpdateProjectInputData

Specifies an array of the **CreateOrUpdateProjectInputData** class that contains information such as the file path, proxy objects, secondary object UIDs.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the new project object UIDs. The response data is based on the class **CreateOrUpdateProjectResponseData**.

getDictionaries API

This API gets dictionary information from Teamcenter.

Inputs

GetSystemSynthesisModelingDataFilter

Specifies an array of the **CreateOrUpdateDictionaryInputData** class to define related data to be downloaded along with the dictionary.

Outputs

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the dictionary information. The response data is based on the class **PrimaryObjectData**.

getTemplateLibraries API

This API retrieves information about template libraries from Teamcenter.

Inputs

GetSystemSynthesisModelingDataFilter

Specifies an array of the **GetSystemSynthesisModelingDataFilter** class elements that defines related data to be downloaded along with the template libraries.

Outputs

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to information about template libraries. The response data is based on the class **PrimaryObjectData**.

getBaseArchitectures API

This API retrieves information about base architectures from Teamcenter.

Inputs

GetSystemSynthesisModelingDataFilter

Specifies an array of the **GetSystemSynthesisModelingDataFilter** class that defines related data to be downloaded along with the base architecture.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to information about the base architecture. The response data is based on the class **PrimaryObjectData**.

getModelContainers API

This API retrieves information about model containers from Teamcenter.

Inputs

GetSystemSynthesisModelingDataFilter

Specifies an array of the **GetSystemSynthesisModelingDataFilter** class that defines related data to be downloaded along with the model containers.

Outputs

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation

- Response data corresponding to information about model containers. The response data is based on the class **PrimaryObjectData**.

getProjects API

This API retrieves information about projects from Teamcenter.

Inputs

GetSystemSynthesisModelingDataFilter

Specifies an array of the **GetSystemSynthesisModelingDataFilter** class that defines related data to be downloaded along with the projects.

Output

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to information about projects. The response data is based on the class **PrimaryObjectData**.

ReviseSystemSynthesisObjects API

This API revises a list of Simcenter System Synthesis objects in Teamcenter.

Inputs

ReviseOrSaveAsInputData

Specifies an array of the **ReviseOrSaveAsInputData** class that contains information such as the new revision ID to revise a Simcenter System Synthesis object.

Outputs

operationStatus

Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the new revision UIDs. The response data is based on the class **SystemSynthesisModelingResponseData**.

saveAsSystemSynthesisObjects API

This API creates copies of a list of Simcenter System Synthesis objects in Teamcenter.

Inputs

vecSaveAsInputs

Specifies an array of the **ReviseOrSaveAsInputData** class that contains information such as new object name and description.

Outputs

operationStatus
Specifies the following outputs based on the **TCMEOperationOutput** class:

- Status of the operation
- Response data corresponding to the new revision UIDs. The response data is based on the class **SystemSynthesisModelingResponseData**.

deleteSystemSynthesisObjects API

This API deletes a list of Simcenter System Synthesis objects in Teamcenter.

Inputs

vecObjectUids
Specifies an array of object UIDs to be deleted.

Outputs

operationStatus
Specifies the status of the operation based on the **TCMEOperationOutput** class:

checkOut API

This API checks out a list of Simcenter System Synthesis objects from Teamcenter.

Inputs

vecObjectUids
Specifies an array of object IDs to be checked out.

comments
(Optional) Specifies any comments for the check-out operation.

changeId
(Optional) Specifies a change ID for the check-out operation.

Outputs

operationStatus
Specifies the operation status based on the **TCMEOperationOutput** class.

checkIn API

This API checks in a list of Simcenter System Synthesis objects in Teamcenter.

Inputs

objectUids
Specifies an array of object IDs to be checked out.

Outputs

operationStatus
Specifies the operation status based on the **TCMEOperationOutput** class.

cancelCheckout API

This API cancels a checkout on a list of Simcenter System Synthesis objects from Teamcenter.

Inputs

ObjectUids
Specifies an array of object IDs whose checkout must be canceled.

Outputs

operationStatus
Specifies the operation status based on the **TCMEOperationOutput** class.

Customizing behavior modeling integration operations by using extensions

Teamcenter provides extension points for the following behavior modeling tool integration operations:

- Save
- Save-as
- Open
- Insert

Extension points are available for the following concurrent modeling operations:

- Import
- Export
- Add to branch
- Update

You can implement these extensions as java programs, tool-specific scripts, or batch files.

Use the following extension points to extend behavior modeling integration:

Extension name	Description
PRE_UI_ACTION	Performs the configured actions before the behavior modeling tool launches Teamcenter dialog boxes. This extension supports only Save and Save-as operations. You can implement this extension using Java only.
PRE_UI_INSERT	Performs the configured actions before the behavior modeling tool launches the Teamcenter insert dialog box. You can implement this extension using Java only.
PRE_CONDITION	Checks the specified conditions before performing the behavior modeling operations. If the **PRE_CONDITION** extension fails, the operation is not performed.
PRE_ACTION	Performs the specified actions before performing the behavior modeling operations.
POST_ACTION	Performs the specified actions after performing the behavior modeling operations.

The **PRE_UI_ACTION** and **PRE_UI_INSERT** extensions take the input as an XML string that is based on the **BHMOperationsSchema.xsd** file. The **PRE_CONDITION**, **PRE_ACTION**, and **POST_ACTION** extensions take the input as an XML string based on the **BHMDesignSchema.xsd** file.

Configuring extensions

Update the *TOOL-NAME*_**BHMIntegrationDefinition.xml** file with information about the extensions. If you are configuring the concurrent modeling extensions, update the **PROJECT_IntegrationDefinition.xml** file.

For example, for integration with MATLAB, you can update the Matlab_BHMIntegrationDefinition.xml file as follows:

```
<Extensions>
 <Extension operationName="SAVE" extensionPoint="PRE_UI_ACTION" >
      <Impl type="JAVA" location="" name="com.teamcenter.matlabcustom.SaveCustom" />
    </Extension>
 <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>
```

- operationName

Specifies the name of the operations that the extension point supports.

The extension points support Save, Save as, Open, and Insert operations.

The PRE_UI_ACTION extension supports only the Save and Save as operations.

- extensionPoint

  Specifies the name of the extension points such as PRE_UI_ACTION, PRE_CONDITION, PRE_ACTION or POST_ACTION.

- implType

  Specifies how the extensions are implemented.

  implType can have the following values:

  o JAVA

    Specifies that the extension is implemented in Java.

    The Java implementation must either implement the **com.teamcenter.behaviormodeling.common.bhminterface.IBHMOperations Extendable** interface or extend the **com.teamcenter.behaviormodeling.common.DefaultBHMOperations Extendable** class.

    The PRE_UI_ACTION extension point supports only Java as the implementation type.

  o BATCH

    Specifies that the extension is implemented as a batch file.

  o SCRIPT

    Specifies that the extension is implemented in the behavior modeling tool-specific script file.

- location

  Specifies the location of the batch file. You must set a value for this variable only when the **implType** is **BATCH**.

- name

  o Specifies the class name when the **implType** is Java.

    The class must be in a JAR file named **customjars** that is located in the **bhm** folder of the Teamcenter installation.

  o Specifies the name of the batch file when the implType is BATCH.

  o Specifies the command ID of the script file when the implType is SCRIPT.

    The script file name corresponding to the command ID is defined in the **MATLAB_BHMConnectorCommands.xml** file.

Configure the PRE_UI_INSERT extension

The PRE_UI_INSERT extension allows you to restrict the search options on the Teamcenter Insert dialog box.

You can implement this extension using JAVA only.

The method for implementing this extension is as follows:

- Implement a custom JAVA class and override the **public BHMExtensionOutput executePreUIAction(String input, Connection connection)** method in this custom class.
- The input string of the **BHMExtensionOutput executePreUIAction** method is an XML file based on the **BHMOperationSchema.xsd** schema.
- Use the **JAXB** utility to generate the XML input strings. This utility is available in the **BHMOperationData** and the **BHMDesign** classes located in the **com.teamcenter.behaviormodeling.common.xml** package.
- The JAXB utility also does the following:
  - Converts the input XML string into a JAVA object.
  - Sets the UID in the JAVA code.
  - Converts the JAVA object back into XML.
  - Sets the XML file as the output in the XML field of the **BHMOperationOutput** object.
- In the input XML file, you can provide two types of UIDs. These UIDs will be used to populate the **Insert** dialog box.
  - The list of UIDs for the insert search operation. In the **Insert** dialog box, you can search using Teamcenter folders or Classification, or you can use Teamcenter search. You can provide UIDs only for one out of the three search options.
  - The list of UIDs for the behavior models that you want to appear in the **Model selected for Insert** list of the **Insert** dialog box.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BHMOperations xmlns="http://www.plmxml.org/Schemas/bhm">
  <options>
    <insertOperationOptions stopOperationOnError="true">
 <!-- List of Teamcenter UIDs for browsing on Insert dialog.
           Enable one of the following: -->
      <BrowseList>
        <TCFolderUIDs>
  <!-- 1. Specifies the Teamcenter folder UIDs -->
          <UID>AkTRBcBQI2ELbD</UID>
          <UID>gQYRBcGLI2ELbD</UID>
        </TCFolderUIDs>
  <!-- 2. Specifies the Classification UIDs -->
        <TCClassUIDs/>
  <!-- 3. Specifies the UIDs available in Teamcenter search dialog-->
        <TCObjectUIDs/>
```

```
      </BrowseList>
    <!-- Specifies the UIDs of objects that will be displayed
              in the Models selected for insert list
    -->
      <SelectList>
        <UID>A6URBk7FI2ELbD</UID>
      </SelectList>
     </insertOperationOptions>
  </options>
</BHMOperations>
```

In the example input XML file:

- UIDs are specified for objects to be displayed in for the **Teamcenter Folder** tab. The **Classification Tree** and **Teamcenter Search** tabs are disabled.

  Only one tab can be enabled at a time.

- UIDs of objects that must always appear in the **Models selected for insert** list are added in the **SelectList** element.

Appendix 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.
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.

Magic Draw objects

The following Magic Draw objects and relations are available once you install the Behavior Modeling integration features:

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

Siemens Industry Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Suites 4301-4302, 43/F
AIA Kowloon Tower, Landmark East
100 How Ming Street
Kwun Tong, Kowloon
Hong Kong
+852 2230 3308

About Siemens PLM Software

Siemens PLM Software, a business unit of the Siemens Industry Automation Division, is a leading global provider of product lifecycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens PLM Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.