TEAMCENTER

Teamcenter Gateway for EDA Integration — Reference

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

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

Support Center: **support.sw.siemens.com**

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

Contents

About this document 1-1

EDA client configurations

About the client configurations 2-1
Naming conventions 2-1
Teamcenter Gateway for EDA configuration templates 2-1
- About the Teamcenter Gateway for EDA configuration templates 2-1
- gatewayCombined_edadef configuration 2-2

Customizing gateway configurations 2-2
RdnAttrMapDefs 2-2

Callbacks

About callbacks 3-1
IntermediateFileStagingDir property 3-1
Teamcenter Gateway for EDA supported callback operations 3-2
- extractCADFromPCB / extractCADFromSchematic 3-2
- extractBOMFromPCB / extractBOMFromSchematic 3-2
- openDesign 3-3
- reOpenDesign 3-4
- closeECAD 3-4
- setDesignReadOnly 3-4
- setDesignCheckedOut 3-5
- prepareVariantInfo 3-5
- generateDesignBOMs 3-6
- generateDesignBOMs2 3-7
- getDesignAttributes 3-8
- updateDesignAttributes 3-8
- updateRelatedDesigns 3-9

Caution while using callbacks supported with common client 3-10
Callback scripts for sharing 3-10

1. About this document

This document provides supplementary information for the ECAD users to develop new or modify existing Teamcenter Gateway for EDA (Electronic Design Automation) integration instances. It is assumed that the user of this document has read Integrating Xpedition and PADS with Teamcenter or Installing and Configuring EDA Gateway for (Non-Siemens EDA) ECAD Applications to install and configure your ECAD tools with Teamcenter EDA.

The Teamcenter Gateway for EDA integration framework allows different ECAD tools to be interfaced with Teamcenter without requiring ECAD tool-specific APIs to support. The customization work involves two parts:

- EDA client configurations
- Callback implementation

This document includes information about managing the EDA data management functionalities for Teamcenter EDA. However, this document does not cover information for EDA part library management, how a viewable is extracted for generating BOM, and about the configurations required to generate derived items and datasets.

2. EDA client configurations

About the client configurations

Every ECAD tool that is integrated with Teamcenter EDA must have its own client configuration. This is required for both Teamcenter EDA embedded and gateway integrations. This document explains the Teamcenter Gateway for EDA integration only.

Naming conventions

The Teamcenter Gateway for EDA client configuration is defined by an XML file that must be located in the *TCEDAECAD_ROOT*/**gateway** folder. The file name must follow the following convention:

*application_name*_**edadef.xml**

The application_name is displayed in the configuration list when you choose **File** > **Configuration** on the EDA Gateway application window.

The following are the COTS design configuration files:

- **expeditionPCB_edadef.xml**
- **expeditionSch_edadef.xml**

Each new gateway configuration is defined by creating a new configuration XML file in the *application_name*_**edadef.xml** folder.

Teamcenter Gateway for EDA configuration templates

About the Teamcenter Gateway for EDA configuration templates

The client configuration must be a valid XML file that is compliant with the supported schema. However, there is no way to support arbitrary values of attributes and elements in the XML file. Therefore, instead of providing the schema directly to users, Teamcenter Gateway for EDA configuration templates are available for ECAD users. The XML template files are located in *TCEDAECAD_ROOT*/*gateway/template*. Users should use these templates as starting configurations to develop their own customized configurations.

The following Teamcenter Gateway for EDA configuration templates are distributed:

- **gatewayCombined_edadef.xml**
- **gatewaySchematic_edadef.xml**

- **gatewayPcb_edadef.xml**
- **gatewaySimulation_edadef.xml**

The selection of the starting template depends on the ECAD tool you use and the data model used to save datasets in Teamcenter.

gatewayCombined_edadef configuration

This configuration allows you save all design data to Teamcenter as one dataset under a CCA item or item revision. The dataset usually contains all related design data such as schematic, simulation, and layout design files, but may contain only one type of the design files.

- **gatewaySchematic_edadef.xml**
- **gatewayPcb_edadef.xml**
- **gatewaySimulation_edadef.xml**

These configurations are generally used together to save schematic, layout designs, and simulation data in separate datasets under the same CCA item or item revision. The layout and simulation datasets are saved directly under the CCA item or item revision. The schematic design dataset is saved under a schematic item or item revision. The schematic item is a child under the BOM view of the CCA item. All datasets saved under the same CCA item/item revision are called related datasets in a family. These configurations can also be used independently to save different types of design datasets in separate CCA item/item revisions.

Customizing gateway configurations

When customizing configurations from the templates, the user must change the fields that are marked as CHANGEME or <fixme>. Other customizations can be made by referring the **EDADesign** section in the Schema Reference help on Support Center.

A gateway configuration must include the **EDAGatewayDef** element. You must not set or change the **className** attribute or the **Preferences** element.

Use the COTS configurations in the *TCEDAECAD_ROOT/gateway* folder as examples.

RdnAttrMapDefs

To pick up the attributes from the BOM files, you must create an additional BOM definition file. This file must be named *application*_**BOMFileDef.xml** and located in the same folder where the client configuration file *application*_**BOMFileDef.xml** is installed. Without the additional definition file, the attributes in the BOM files are not picked up and sent to common client for occurrence mapping.

Note:

Refer to the **EDAClientDefSchema** section in the Schema Reference help on Support Center for more information about this element. However, how to generate viewable is out of the scope of this document.

Here is an example file:

```
<?xml version="1.0" encoding="UTF-8"?>
<EDABOMFileDef>
    <AttrColumn columnNumber="1" columnHeading="PART_NUMBER" />
    <AttrColumn columnNumber="2" columnHeading="RDN" />
    <AttrColumn columnNumber="3" columnHeading="PART_NAME" />
    <AttrColumn columnNumber="6" columnHeading="CALL_NAME" />
    <AttrColumn columnNumber="5" columnHeading="CALL_NUMBER" />
</EDABOMFileDef>
```

This example shows the basic configuration elements:

- One root element **EDABOMFileDef** is required.
- A number of **AttrColumn** elements are required to define associations of column numbers and headings.

The order of the elements can be arbitrary. The heading is the name of the attribute to be picked up from the BOM files (see the **generateDesignBOMs** callback section). If the attributes listed in this definition file are not in the **RdnAttrMapDefs**, they are ignored for mapping to any Teamcenter attributes or occurrence notes.

3. Callbacks

About callbacks

Callbacks are a key part of the Teamcenter Gateway for EDA integration. Although none of the supported callbacks is absolutely required, how tightly an ECAD tool is integrated using Teamcenter Gateway for EDA integration framework depends on how well the callbacks are developed.

The Teamcenter Gateway for EDA integration framework supports a limited number of callback operations specially designed for Teamcenter Gateway for EDA integration. They are all listed in the templates accordingly and defined as script types. Each template includes only the callback operations that can be used by the model for which the template is configured. In customization, not every callback operation listed in the templates is required. Only those that have real implementations should be included. Others can be deleted or simply commented out. Each callback definition must specify the names of the operation and command. The customization cannot change the operation name in the templates. The real command name should replace the <**fixme**> or <**CHANGEME**> field in the template for the associated operation. The commands must be supported by real executable functions such as command scripts or java methods.

For example,

```
<callback operation="setDesignReadOnly" type="script"
          command="set_design_folder_read_only.bat" />
```

Where <**fixme**> or <**CHANGEME**> is replaced by **set_design_folder_read_only.bat** and is an executable command script and must be found in the path that includes the gateway folder.

For additional information on how to develop Java callback, refer to the Callbacks in Teamcenter EDA help.

IntermediateFileStagingDir property

This property defines a location where intermediate CAD files are stored. The intermediate CAD files, which are used to generate viewable datasets in Teamcenter, can be manually copied into the **IntermediateFileStagingDir** or automatically saved by the extracting the CAD file callback. In general, the **IntermediateFileStagingDir** should be in the system temp directory. If the **IntermediateFileStagingDir** is not specified, or if it is empty, the user will be asked to select the intermediate CAD files manually with an **xfatf/xsch** generation dialog box during Teamcenter **Save** or **Save As** operations.

Teamcenter Gateway for EDA supported callback operations

extractCADFromPCB / extractCADFromSchematic

This is called whenever viewable dataset is required in the save operations. The callback must extract intermediate CAD files from the current design and saves the intermediate CAD files into **IntermediateFileStagingDir** mentioned before. Therefore, the **IntermediateFileStagingDir** must be defined in the **TCEDAClient.properties** file whenever this callback is defined. Different ECAD tools generate intermediate CAD files differently. The generated intermediate CAD files must meet the requirement of the translator to generate the viewable.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with two arguments:

- **designFolder**
- **cadFileFolder**

The **designFolder** is the absolute path of the working folder containing the current design. The **cadFileFolder** is the path of destination folder where all extracted CAD files are saved after the callback is finished. It is the same folder as what is defined by the **IntermediateFileStagingDir** property. However, the callback command script should only use this passed argument as the input. Do not use the hard-coded path directly from what is defined by the **IntermediateFileStagingDir** property.

If this callback is not defined but the **IntermediateFileStagingDir** is defined and valid, the user may manually copy CAD files to **IntermediateFileStagingDir** so that the EDA client will not ask for the CAD files during Teamcenter save operations.

extractCADFromPCB is called for layout (PCB) designs while **extractCADFromSchematic** is called for schematic designs.

extractBOMFromPCB / extractBOMFromSchematic

This callback extracts BOM from the current ECAD design. By default, Teamcenter Gateway for EDA integration framework uses viewable to generate BOM. However, this callback overrides the default.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with two arguments:

- **designFolder**
- **edabomFile**

The **designFolder** is the absolute path of the working folder containing the current design. The **edabomFile** is the path of the destination file where all extracted component information is saved. The caller only provides the name of **edabomFile**. It is the callback responsibility to create/update the file

with component information extracted from the design. The **edabomFile** must be an XML file compliant to the **EDADesignSchema.xsd** with only component elements. Here is a simple edabomFile example to follow:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="tomcounter10" bom="true">
 <dataset type=”schematic” />
 <component name="7404" itemId="7404" revId="-" quantity="1">
 <rdn name="U6" />
 </component>
 <component name="74161" itemId="74161" revId="-" quantity="2">
 <rdn name="U1" />
 <rdn name="U2" />
 </component>
 <component name="74185A" itemId="74185A" revId="-" quantity="3">
 <rdn name="U3" />
 <rdn name="U4" />
 <rdn name="U5" />
 </component>
 </CCA>
</EDADesign>
```

In the example, each component is described as an item in Teamcenter. You must provide its **name**, **item ID**, **item revision** (revId), quantity and reference (rdn) designators. The **itemId** must be a unique ID for the component in Teamcenter. The value “-“ in the **revId** attribute indicates the latest revision but you can use other values for specific revisions.

The element dataset is a dummy dataset required by the schema. Only the *type* attribute is required to specify the primary data type, which can be schematic or PCB. **extractBOMFromSchematic** is called for schematic designs while **extractBOMFromPCB** is called for layout (PCB) designs.

openDesign

This callback is used to start the ECAD tool and opens the current design in the tool.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with a single argument:

- **designFolder**

The callback must find out which file in the folder should be opened by the tool.

This callback is only called after the Teamcenter open operation is finished. Any error during this callback execution will not impact the Teamcenter Open operation. If the user does not want any ECAD tool to be started, delete this callback definition from the configuration.

This callback is only used when opening a known design in Teamcenter Gateway for EDA (double-clicking a design in the **Known Designs** tab). In addition, the Teamcenter Gateway Open operation (to open a design not already in the staging directory) uses the **reOpenDesign** callback, which should only open the design if the ECAD tool has already been started. This change makes the Teamcenter open operation consistent with other operations such as refresh and checkout.

Refer to the notes about **enableOpenRelatedDatasets** configuration for additional information.

reOpenDesign

This callback is used to reopen the current design in the ECAD tool that is already opened.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with a single argument:

- **designFolder**

This callback is called whenever the existing design copy in the staging directory is updated by a Teamcenter operation such as refresh to the latest, save-as copy, and checkout with refresh. This function is required to replace the obsolete design copy previously opened in the ECAD tool. If the ECAD tool is not accessed, this callback command will not be executed. However, if the ECAD tool is started with the obsolete design, this callback command must be executed to make sure no obsolete design is used. The callback is executed on its own depending on whether the ECAD tool is accessed or not. Any error during this callback execution will not impact the Teamcenter operations that have already updated the design files in the staging directory.

The callback will search which file in the folder should be reopened by the tool.

closeECAD

This callback closes the ECAD tool. No input argument is passed to the callback command.

This callback is called before the Teamcenter purge cache operation begins. This is required to unlock the design files and prevent the active ECAD tool from being hung or crashed, potentially caused by deleting the opened design from the staging directory. The error during this callback execution will not impact the subsequent Teamcenter purge cache operations. However, manual cleanup and ECAD tool termination may possibly be required depending on nature of the errors. The user can choose to manually close the ECAD tool instead of to define and implement this callback.

setDesignReadOnly

This callback sets the current design to read-only.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with a single argument:

- **designFolder**

The callback is called after the Teamcenter operations that change the design lock status to checked-in. Such operations include check-in, open for reference, save, and save-as with the check-in option selected.

setDesignCheckedOut

This callback sets the current design to writable state.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with a single argument:

- **designFolder**

This callback is called after the Teamcenter operations that change the design lock status to checked-out. Such operations include check-out, open for modification, save, and save-as without the check-in option selected.

prepareVariantInfo

This callback automates the generation of variant names for the design to be saved. Teamcenter Gateway for EDA uses a file-based interface to pass information between ECAD tools and Common Client. To support saving designs with variants, Teamcenter Gateway for EDA framework uses a default subfolder named **edabom** for passing variant design information to EDA Common Client. The Teamcenter EDA data model assumes that each design variant is specified by its BOM.

The subfolder **edabom** must be directly under the design folder. Each file in this folder is a BOM report that represents either a variant or the base design. The file names must follow the following conventions:

- *<variant name>***_var.csv**

  for variant BOM

- *<design name>***_base.csv**

  for base design

The Teamcenter Gateway for EDA integration framework calls the command of this operation with a single argument:

- **designFolder**

This callback must generate all required files under the *designFolder/edabom* folder. If the **edabom** subfolder does not exist, the callback must also create the subfolder. The **.csv** file type is the only file

type that is supported currently. If this callback is not defined, the user must manually generate or copy the subfolder and files required for representing a design with variants.

generateDesignBOMs

This callback automates the generation of all BOMs of variants and base design. The execution of this callback updates the BOM files generated by the **prepareVaraintInfo** callback with complete component information. Not all BOM files are required to be updated. If the **prepareVaraintInfo** callback can generate all BOM files with complete component information, this callback is not required. However, it is less efficient to generate all BOM information by **prepareVaraintInfo** callback because some BOMs in the list may not be required. A BOM file is required to be updated only when its corresponding BOM is selected by the user in the pre-save dialog box.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with the following arguments:

- **designFolder**
- **true** or **false**
- list of variant names

The first argument passes the full path of the selected design folder so that the callback finds where the **design** and **edabom** subfolders are located. The second argument tells whether the base BOM is required or not. If it is true, then the **<design name>_base.csv** file must be updated to list all components included in the base design. The callback can ignore this argument if the client is configured to use **ExtractBOMFrom<XXX>** callback for base design BOM extraction. The remaining arguments are optional to list names of variants. This name list works as a filter to indicate which variant BOM files in the **edabom** folder are required to be updated with component list. The execution of callback may ignore this optional list and generate all variant BOMs and base design BOM files in the **edabom** subfolder. The Gateway framework can pick the BOM files from them selectively. However, updating only the required BOM files is more efficient. If this callback is not defined, it is the user's responsibility to manually update the BOM files to make sure the correct components are in the BOM files.

The Gateway framework supports the **.csv** BOM file format only. Therefore, BOM files should follow the conventions as below:

- Base BOM report is contained in **<design name>_base.csv**
- Variant BOM report is contained in **<variant name>_var.csv**
- The first line of the CSV file contains all attribute names separated by commas:

```
PART_NUMBER,RDN[,PART_NAME][,REV_ID][, <attribute 1 name>[,...
[,<attribute n name>]...]]
```

- Remaining lines of the CSV file contain all attribute values separated by commas:

```
<PartNumber>,<rdn>[,<PartName>][,<revId>][, <attribute 1 value>[,...
[,<attribute n value>]...]]
```

- The first four columns (separated by commas) must be ordered as, **PART_NUMBER**, **RDN** (reference designator), **PART_NAME**, and **REV_ID** (revision ID). The first two columns are required and the remaining columns are optional. However, if other attributes are included, the **PART_NAME** and **REV_ID** columns must be included. The empty columns separated by commas are acceptable for the optional attributes.

- All lines must have the same number of *cells*, that is a fixed number of columns.

Here is an example of supported **.csv** BOM file containing three lines:

```
PART_NUMBER,RDN,PART_NAME,REV_ID,VALUE,COST
R-2w-1234,R1,resister,”-“,2K,$0.10
GCOM-74LS04,U1,74LS04,”-“,74LS04,$0.50
```

generateDesignBOMs2

This callback is similar to previously described **generateDesignBOMs**. It is provided to support integration of combined models, as well as dual models. It will be called if **generateDesignBOMs** is not defined in the configuration.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with the following arguments:

- designFolder
- base BOM option
- variant BOM option
- list of variant names

Comparing with generateDesignBOMs, the previous second argument is replaced with two arguments. These two arguments pass the user selections for base and variant BOMs respectively. For combined model, the valid options are:

- **WithoutBOM**
- **FromPCB**
- **FromSchematic**

For dual model, the valid options are:

- **WithoutBOM**
- **WithBOM**

The callback must generate the BOMs according to the user selected options.

getDesignAttributes

This callback collects ECAD design attributes that may be saved together with the primary design dataset to Teamcenter using proper attribute mapping settings. If it is not defined, no design attributes can be saved to Teamcenter.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with the following arguments:

- **designFolder**
- **dataType**
- **attributeFileName**

The **designFolder** is where the primary design data are located. The **dataType** can be one of the following:

- **pcb**
- **schematic**
- **simulation**
- **combined**

The argument **attributeFileName** specifies the name of the file where the attribute name and value pairs should be written by the callback implementation. Each line of the file contains one pair of the attribute name and value. The name and value can be separated with either **=** or a **blank space**. No blank space is allowed in the attribute names.

updateDesignAttributes

This callback is used to update the design dataset after the dataset is downloaded to the staging directory. It is called to maintain Teamcenter information as well as the ECAD design properties.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with the following arguments:

- **designFolder**
- **dataType**
- **checkedOut**
- **datasetUID**
- **attributeFileName**

The first two arguments are defined the same way as what is described in the **getDesignAttributes** callback.

The argument **checkedOut** is either true or false to indicate whether the dataset has a checked out lock in Teamcenter or not. The **datasetUID** is the Teamcenter UID for the dataset. These two arguments can be optionally cached locally with the design files as references.

The argument **attributeFileName** specifies the name of the file where the callback should read the attributes passed from Teamcenter. Each line of the file contains one attribute. The name and value of the attribute are separated by **=**. No blank space is allowed in the attribute name.

updateRelatedDesigns

This callback supports downloading related design datasets. It is optionally required only when the **enableOpenRelatedDatasets** attribute in the configuration is set to true.

The Teamcenter Gateway for EDA integration framework calls the command of this operation with the following arguments:

- **edaDesignFileName**
- **edaStatusFileName**

The argument **edaDesignFileName** specifies the name of the file that contains the information of all datasets downloaded to the staging directory. The argument **edaStatusFileName** specifies the name of the file that contains the status information of the last downloading operation.

The callback implementation should check the status first to make sure there was no error in the last operation and then get all related dataset information from the EDA design **.xml** file named **edaStatusFileNameedaDesignFileName**. For each related dataset, the callback can repeat what to do by the **updateDesignAttributes** callback with a different dataType. The details about EDA design and status **.xml** files can be found in the example folder under the installed EDA root.

Although this callback is primarily for related design datasets, it is very similar to the **postOpen** and **postRefresh** callbacks supported by the common client. This is because it is called after the same Teamcenter operations and the passed arguments contain all information including both

primary and related datasets. You can use **updateDesignAttributes** for the primary dataset and **updateRelatedDesigns** for the related designs separately, or you can use **updateRelatedDesigns** alone for both primary and related datasets.

Caution while using callbacks supported with common client

The Teamcenter Gateway for EDA framework inherits all callbacks from common client, including those for pre- and post- operation callbacks and derived item and dataset configurations. How to configure the generation of derived items and datasets is out of scope of this document. Detailed information can be found in the example folder under the installed EDA root directory. The distributed Teamcenter Gateway for EDA templates do not include the pre and post operation callbacks. Therefore, they must be added during the customization based on requirements. Each callback can only be defined once in an EDA client configuration file.

The Teamcenter Gateway for EDA supported callbacks are functional while the common client supported pre- and post- operation callbacks are operational. The Teamcenter Gateway for EDA framework calls a functional callback in multiple operations, while the common client calls an operational callback only once in each operation. Hence, there must be consistency between different operational callbacks if a same function or task is required by different operations.

For example, the **postCheckIn** callback can include the task that **setDesignReadOnly** callback does. However, for maintaining consistency, other post operation callbacks such as, **postCancel** should also include the task that **setDesignReadOnly** callback does. The callback developer must provide a set of callbacks that do not conflict with each other.

Callback scripts for sharing

The following two COTS callbacks for the Teamcenter Gateway for EDA integration framework are delivered for sharing:

- **set_design_folder_read_only.bat**
- **set_design_folder_writable.bat**

An ECAD design contains files and subfolders under a single design folder. The above listed sharable callbacks set read-only or write access on all files and subfolders along with the design folders. These settings meet most requirements for the ECAD tool in general. However, setting the design to read-only is not equivalent to setting all files and subfolders to read-only. Different ECAD tools and different customers may have different requirements to lock and unlock the designs for modification. For example, some ECAD tools require all temporary files to have write access at all times so that the design can be opened for modification when required. For such situations, you must use the COTS shareable callbacks with caution.