TEAMCENTER

Teamcenter EDA Design Connector 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

Supported integration types 2-1

Supported configuration types 3-1

Overview of creating a new integration 4-1

Embedded Teamcenter EDA operations
Overview of Teamcenter EDA embedded operations 5-1
Save As 5-2
Revise 5-4
Save 5-5
Check-In 5-7
Save Derived Data 5-9
Open 5-10
Check-Out 5-11
Cancel Check-Out 5-12
Refresh 5-13
Purge Working Files 5-14
Item Info 5-15
Design Info 5-15
Logout 5-15
BOM Compare 5-16
Issue Manager 5-17
Collaboration 5-17
Workflow 5-18
Inbox 5-20

About configuration file 6-1

EDADesign file 7-1

EDAStatus file 8-1

Common Client API
Overview of common client APIs 9-1
setApplicationName 9-2
preSaveAs2 9-2

saveAs2 9-3
preRevise 9-4
revise 9-5
preSave 9-6
save 9-7
preCheckIn 9-8
checkIn 9-9
saveDerivedData 9-10
open 9-11
checkOut 9-12
cancelCheckOut 9-12
refresh 9-13
purgeCache 9-14
itemInfo 9-15
showDesignInfo 9-15
getStagingDir 9-16
setPreferencesByFile 9-17
getPreferences 9-18
logout 9-19
logoutNoConform 9-19
reconcilePartsSelect 9-20
reconcileParts 9-21
initiateIssueReport 9-22
initiateCollaborationIssue 9-23
publishCollaborationFiles 9-24
downloadCollaborationFiles 9-25
initiateDesignWorkflow 9-25
initiatePartWorkflow 9-26
checkPartWorkflowStatus 9-27
getMyWorklist 9-28
progress 9-29
downloadBOM 9-29
copyDesignFolder 9-30
custom <operationName> 9-31

1. About this document

This document provides supplementary information for the developers to create new or modify existing Teamcenter EDA (Electronic Design Automation) design connector integration instances. It is assumed that the user of this document has read Xpedition and PADS Integration With Teamcenter or Teamcenter EDA Gateway for (Non-Siemens EDA) ECAD Applications and understands the basic operations of EDA.

The EDA integration connector framework allows different ECAD tools to be interfaced with Teamcenter using ECAD tool-specific APIs to support an embedded look and feel. The customization work involves three parts:

- ECAD tool configuration
- EDA client configuration
- ECAD tool coding to interface to EDA common client

This help contains instructions for customizations for the EDA design management functionalities for Teamcenter EDA only.

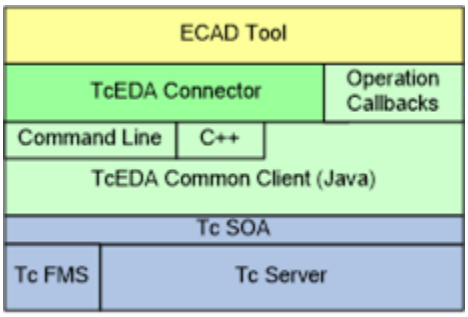
2. Supported integration types

There are two methods of integrating within EDA. The traditional embedded integration enables an ECAD-tool centric integration, while the gateway integration enables a generalized integration without requiring specific API supports.

Embedded Integration

An embedded integration embeds a Teamcenter menu, or some other UI model, within the ECAD tool itself using vendor provided tools. This Teamcenter menu contains individual menu selections to perform operations such as open, save, checkin, or checkout designs to and from Teamcenter.

Embedded ECAD integrations pass data and status back and forth with the EDA common client via an EDADesign file and EDAStatus XML files. The EDADesign file contains the information about the ECAD design to save or retrieve from Teamcenter. EDA common client provides three sets of interfaces for embedded ECAD tool integrations.



- C API should be used by ECAD tools that can invoke C functions.
- Command line API can be used by ECAD tools that can only use scripts.
- Java

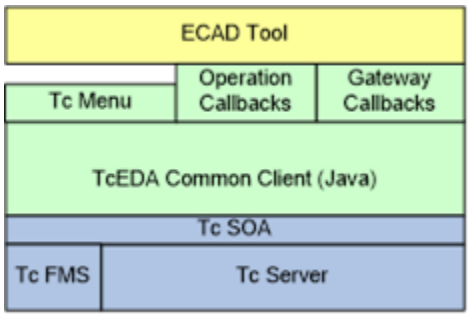
EDA common client contains the common business logic for all ECAD integrations. It is responsible for:

- Reading the client configuration file and acting appropriately on it's contents.
- Displaying user interface dialog boxes to gather user input, and providing Teamcenter navigation and search.
- Reading the EDADesign file and calling SOA to import data into Teamcenter.
- Calling SOA to get data from Teamcenter and writing an EDADesign file.

- Zipping and uploading ECAD design files to Teamcenter, downloading and unzipping ECAD design files from Teamcenter to the EDA staging directory.
- Caching Teamcenter objects in EDA Client cache for quick meta-data access.
- Launching ECAD tool that is related to a selected Teamcenter EDA design object.

Gateway Integration

A gateway integration does not embed anything within the ECAD tool. All interactions with Teamcenter are done through the **eda_gateway** application, which is a separate application from both the ECAD tool and Teamcenter. This document is intended for creating embedded integrations. If you want to create a gateway integration, refer to the document TcEDAGatewayIntegrationGuide.doc in the *%TCEDAECAD_ROOT%\example\docs* folder.



3. Supported configuration types

There are two generic configuration types supported by the EDA common-client.

Dual-Model

The dual-model configuration type is the most common form, and is suggested wherever possible. This model is used for ECAD tools which use separate tools and storage for each design type (PCB and Schematic initially – thus dual, but now adding Simulation). In this model, you would create separate configuration files for each design type. This allows saving different Datasets in Teamcenter (for each tool), under the same CCA for concurrent engineering work.

Combined-Model

The combined-model configuration should only be used when necessary, in the case where the ECAD tool integration makes no distinction between the various design types. For instance, in Mentor BoardStation the EDA connector is built into the Design Manager, and all design files are stored under a common design folder. Thus all of these design files are then saved to Teamcenter within a single Dataset.

4. Overview of creating a new integration

In order to create a completely new integration, there are several tasks you must perform.

Configuration File(s)

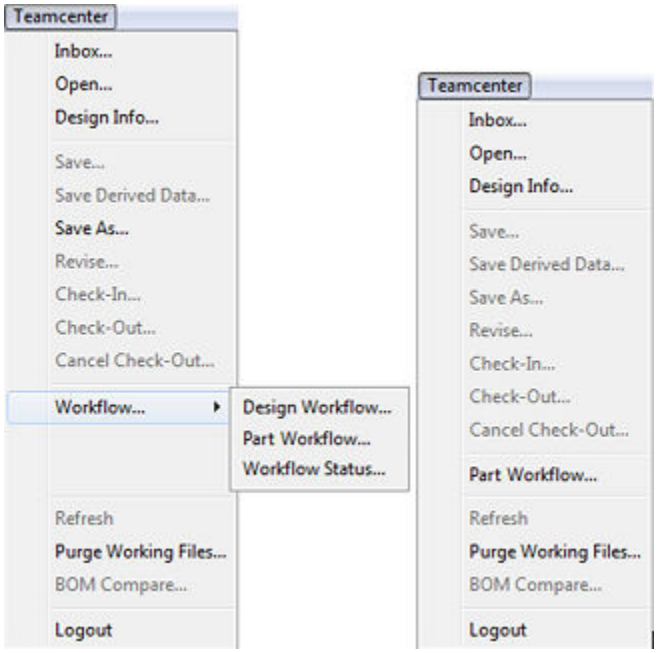
Create configuration file(s) for each ECAD tool you wish to integrate. See *Configuration File* for a full description of configuration files.

For a dual-model integration create two or three configuration files, one for the PCB tool, one for the schematic tool, and optionally one for the simulation tool.

For a combined-model integration create a single configuration file.

Embedded Code

Create code to embed a menu or command buttons, and underlying code, into the ECAD tool(s). The menu/dialog should be laid out similar to this and in the same order:



The menu on the left is our default menu, to be used when Active Workspace (AW) is not used by the customer. The Design Workflow and Workflow Status menu commands raise standard dialogs.

The menu on the right can be used by customers who are using AW. The Design Workflow and Workflow status commands are removed, as those actions can be performed on the Design Info AW page.

Each menu/command should run embedded code to perform the steps documented in the next section, Embedded Teamcenter EDA Operations. This code will need to read and write EDADesign.xml files. The code will also need to read (but not write) EDAStatus.xml files. Both EDADesign.xml and EDAStatus.xml files are fully documented in the Teamcenter EDA Schema References document.

Create Dataset Type

Your configuration should be specifying a new Dataset type unique to your integration. You must use Business Modeler IDE (BMIDE) to create a custom template to define your Dataset type(s) in the Teamcenter server. See *edaserver* for more details.

Create License and Validate

If the licenseKey is not specified in the configuration, then the applicationName specified must map to a valid Teamcenter license value. See *Licensing* for details.

Dataset Images

Your new Dataset type will use the default image (gear) in Rich-Client, Thin-Client, and the EDA common-client dialogs by default. Therefore, you should create new unique images for them. See *Dataset Type Images* for more details.

Supporting Open From Rich-Client

Most of the integration is triggered from your embedded code. However, Teamcenter Rich-Client has a feature to support opening a Dataset's contents within the appropriate application. This is enabled via registry modification and a combination of script/batch files (see *Rich-Client*).

5. Embedded Teamcenter EDA operations

Overview of Teamcenter EDA embedded operations

A new embedded EDA connector should insert a Teamcenter menu in the menu bar of the ECAD tool, and add a menu selection for each EDA operation. If integrating into the tools menu is not possible, investigate adding a popup dialog with menu push-buttons, etc. Each of these menu items should trigger custom connector code to support the following EDA operations:

- **Save As**
- **Revise**
- **Save**
- **Check In**
- **Save Derived Data**
- **Open**
- **Check Out**
- **Cancel Check Out**
- **Refresh**
- **Purge Working Files**
- **Item Info**
- **Design Info**
- **Logout**
- **BOM Compare**
- **Issue Manager**
- **Collaboration**
- **Workflow**
- **Inbox**

Save As

This operation allows the user to save a new design or save a copy of an existing design. The connector code should perform the following steps to implement this operation:

1. Create an EDADesign XML file (see **EDADesign file example**) with preSaveAs2 info. This EDADesign file does not need BOM information, but must contain variant definitions if variants exist.

2. Call **–preSaveAs2** API.

3. Check status file returned from call.

    a. Abort if status != “Success”, display the content of the Message text field to the user.

    b. Check the saveAsCopy status. If it is **true** then get the designFolder status as well. Based on the configuration file setting of **copyFilesInSaveAsCopy** do the following:

    - If *copyFilesInSaveAsCopy* = “true”, then the design was copied during the preSaveAs2 call. Use the designFolder value to open the copied design and set working directory appropriately.

    - If *copyFilesInSaveAsCopy* = “false”, then you need to copy the design here in your connector code. After copying it, open that copy and set working directory appropriately.

    c. If changed, update the itemId and itemName in the design from values in the status file.

    d. If *BOMOption* indicates BOM is requested, extract the component list from the design. This will be used to create component elements in a new **EDADesign** file (in step 5).

    e. If *viewableOption* indicates a viewable is requested, create the appropriate intermediate viewable files in a temporary directory. Include a data set element (specifying this temp folder) for the viewable in the new **EDADesign** file (in step 5).

    f. The *datasetMappedAttrNames* attribute contains a comma-separated list of the names of the CAD design attributes to be included in the EDADesign file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the **EDADesign** file passed to saveAs2, any that are not in this list will be ignored.

    g. The *rdnMappedAttrNames* attribute contains a comma-separated list of the names of the CAD component occurrence attributes to be included in the EDADesign file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to saveAs2, any that are not in this list will be ignored.

    h. Optionally check the **DerivedDataset** element(s) to ensure that the required files are generated .

i. The *VariantBomOption* elements contain the variantName and BOMOption for each variant to be saved. Extract BOM as requested for each variant, and include in the **EDADesign** file passed to saveAs2.

j. For each *VariantBomOption* element, optionally check the DerivedDataset sub-element(s) to ensure that the required files are generated.

k. If the *checkIn* attribute is “true”, you may want to mark the design as read only if possible, or some other action within the ECAD tool context.

l. If the *doNotKeepFlag* attribute is “true”, close the current design and change the working directory permissions so that the common client can delete the design files and folders.

m. The *variantMappedAttrNames* attribute contains a comma-separated list of the names of the CAD variant attributes to be included in the EDADesign file passed to the **–saveAs2** API. You can ignore this list and send all attributes in the **EDADesign** file passed to saveAs2. The common client will ignore the attributes that are not specified in this list.

4. Save the design, and any other information that may be needed to reopen this from Teamcenter in the future, to the local file system. Some ECAD systems may require the design to also be closed.

5. Create a new EDADesign XML file (see **EDADesign file example**) with saveAs2 info. This file must contain BOM information if requested, as well as other information that may have been requested, see above.

6. Call **–saveAs2** API.

7. Check status file returned from call.

   a. Abort if status != “Success”, display to the user the content of the “Message” text field.

   b. Save the Teamcenter unique identifier, UID, locally to be used to identify this design in future EDA operations.

   c. If checkIn = **true** set the design to read-only if possible, and save this status to be tested before future EDA operations.

8. Read EDADesign file returned from call.

   a. If *updateComponentOccurrenceInfo* is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute “UID”.

9. Reopen the design if it was closed and not deleted.

Revise

This operation allows the user to revise a new design in Teamcenter. The connector code should perform the following steps to implement this operation:

1. Create an **EDADesign** XML file with **preRevise** info. The **EDADesign** file must contain variant definitions if variants exist. BOM information should be provided to support the BOM Compare page in the Revise dialog, however it is not required.

2. Call **–preRevise** API.

3. Check status file returned from call.

    a. Abort if *status* != **Success**, display the content of the Message text field to the user.

    b. If *BOMOption* indicates BOM is requested, extract the component list from the design. This is required to create component elements in a new EDADesign file (Step 5).

    c. If *viewableOption* indicates a viewable is requested, create the appropriate intermediate viewable files in a temporary directory. Include a **DataSet** element (specifying this temp folder) for the viewable in the new EDADesign file (in step 5).

    d. The *datasetMappedAttrNames* attribute contains a comma-separated list of the names of the CAD design attributes to be included in the EDADesign file passed to the **–revise** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to revise, any that are not in this list will be ignored.

    e. The *rdnMappedAttrNames* attribute contains a comma-separated list of the names of the CAD component occurrence attributes to be included in the EDADesign file passed to the **–revise** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to revise, any that are not in this list will be ignored.

    f. Optionally check the **DerivedDataset** element to ensure that the required files are generated.

    g. The *VariantBomOption* element contain the *variantName* and *BOMOption* for each variant to be saved. Extract BOM as requested for each variant, and include in the EDADesign file passed to revise.

    h. For each *VariantBomOption* element, optionally check the **DerivedDataset** sub-elements to ensure that the required files are generated.

    i. If the *checkIn* attribute is “true”, you may want to mark the design as read-only if possible, or some other action within the ECAD tool context.

    j. If the *doNotKeepFlag* attribute is “true”, close the current design and change the working directory permissions so that the common-client can delete the design files and folders.

k. The *variantMappedAttrNames* attribute contains a comma-separated list of the names of the CAD variant attributes to be included in the **EDADesign** file passed to the **–revise** API. You may optionally ignore this list and just send all attributes in the **EDADesign** file passed to revise. The common client ignores the attributes that are not specified in this list.

4. Save the design, and any other information that may be needed to reopen this from Teamcenter in the future, to the local file system. Some ECAD systems may require the design to be closed.

5. Create a new EDADesign XML file (see **EDADesign** file example) with **saveAs2** info. This file must contain BOM information if requested, as well as other information that may have been requested, see above.

6. Call **–revise** API.

7. Check status file returned from call.

   a. Abort if *status* != **Success**, display to the user the content of the **Message** text field.

   b. Save the Teamcenter unique identifier, *UID,* locally to be used to identify this design in future EDA operations.

   c. If *checkIn* = **true** set the design to read-only if possible, and save this status to be tested before future EDA operations.

8. Read **EDADesign** file returned from call.

   a. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute “UID”.

9. Reopen the design if it was closed.

Save

This operation allows the user to save a new version of the existing design. The checkout lock will be retained after the save. The connector code must perform the following steps to implement this operation:

1. Validate that you have a UID for this design. Optionally, if there is no UID, switch to **Save As...** operation

2. Optionally, validate that the design is currently checked out to the current user.

3. Create an EDADesign XML file (see **EDADesign file example**) with preSave info. The EDADesign file must contain variant definitions if variants exist. BOM information should be provided to support the BOM Compare page in the Save dialog, however it is not required.

4. Call **–preSave** API.

5. Check status file returned from call.

    a. If *status* != **Success**, check DesignTcStatus == **tc_unknown**. If it is true, switch to **Save As...** operation, otherwise, abort the operation and display the content of the message text field to the user.

    b. If *BOMOption* indicates BOM is requested, extract the component list from the design. This will be used to create component elements in a new EDADesign file (Step 7).

    c. If *viewableOption* indicates a viewable is requested, create the appropriate intermediate viewable files in a temporary directory. Include a DataSet element (specifying this temp folder) for the viewable in the new EDADesign file (in step 7).

    d. The *datasetMappedAttrNames* attribute contains a comma-separated list of the names of the CAD design attributes requested to be included in the **EDADesign** file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the **EDADesign** file passed to **–saveAs2**, any that are not in this list will be ignored.

    e. The *rdnMappedAttrNames* attribute contains a comma-separated list of the names of the CAD component occurrence attributes requested to be included in the **EDADesign** file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the **EDADesign** file passed to **–saveAs2**, any that are not in this list will be ignored.

    f. Optionally check the **DerivedDataset** elements to ensure that the required files are generated.

    g. The *VariantBomOption* elements contain the *variantName* and *BOMOption* for each variant to be saved. Extract BOM as requested for each variant, and include in the EDADesign file passed to save.

    h. For each *VariantBomOption* element, optionally check the **DerivedDataset** sub-elements to ensure that the required files are generated.

    i. The *variantMappedAttrNames* attribute contains a comma-separated list of the names of the CAD variant attributes to be included in the **EDADesign** file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to **saveAs2**. The common client will ignore the attributes that are not specified in this list.

6. Save the design, and any other information that may be needed to reopen this from Teamcenter in the future, to the local file system. Some ECAD systems may require the design to also be closed.

7. Create a new EDADesign XML file (see **EDADesign file example**) with save info. This file must contain BOM information if requested, as well as other information that may have been requested, see above.

8. Call **–save** API.

9. Check status file returned from call.

    a. Abort if *status* != **Success**, display to the user the content of the Message text field.

    b. Save the *UID* locally to be used to identify this design in future EDA operations.

10. Read EDADesign file returned from call.

    a. If **updateComponentOccurrenceInfo** is true in configuration then **EDADesign** file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute UID.

    b. If **updateTcEdaUID** is true then ECAD tool connector needs to update TcEdaUID property for ECAD components in ECAD design using rdn components attribute information.

11. Reopen the design if it was closed.

Check-In

This operation allows the user to checkin a new version of the existing design. The checkout lock will be removed after the checkin. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Optionally, validate that the design is currently checked out.

3. Create an **EDADesign** XML file (see **EDADesign file example**) with **preCheckin** info. The **EDADesign** file must contain variant definitions if variants exist. BOM information should be provided to support the BOM Compare page in the **Check-In** dialog, however it is not required.

4. Call preCheckin API.

5. Check status file returned from call.

    a. Abort if *status* != **Success**, display the content of the Message text field to the user.

    b. If *BOMOption* indicates BOM is requested, extract the component list from the design. This will be used to create *Component* elements in a new EDADesign file (in step 7).

c. If *viewableOption* indicates a viewable is requested, create the appropriate intermediate viewable files in a temporary directory. Include a *DataSet* element (specifying this temp folder) for the viewable in the EDADesign file (in step 7).

d. The *datasetMappedAttrNames* attribute contains a comma-separated list of the names of the CAD design attributes requested to be included in the EDADesign file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to saveAs2, any that are not in this list will be ignored.

e. The *rdnMappedAttrNames* attribute contains a comma-separated list of the names of the CAD component occurrence attributes requested to be included in the EDADesign file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to saveAs2, any that are not in this list will be ignored.

f. Optionally check the **DerivedDataset** elements to ensure that the required files are generated (see What`s new for TC EDA2.4).

g. The *VariantBomOption* elements contain the *variantName* and *BOMOption* for each variant to be saved. Extract BOM as requested for each variant, and include in the EDADesign file passed to checkIn.

h. For each *VariantBomOption* element, optionally check the **DerivedDataset** sub-elements to ensure that the required files are generated (see What`s new for TC EDA2.4).

i. If the *doNotKeepFlag* attribute is “true”, close the current design and change the working directory permissions so that the common-client can delete the design files and folders.

j. The *variantMappedAttrNames* attribute contains a comma-separated list of the names of the CAD variant attributes requested to be included in the EDADesign file passed to the **–saveAs2** API. You may optionally ignore this list and just send all attributes in the EDADesign file passed to **saveAs2**. The Common client will ignore the attributes that are not specified in this list.

6. Save the design, and any other information that may be needed to reopen this from Teamcenter in the future, to the local file system. Some ECAD systems may also require the design to be closed.

7. Create a new EDADesign XML file (see **EDADesign** file example) with checkIn info. This file must contain BOM information if requested, as well as other information that may have been requested.

8. Call **–checkIn** API.

9. Check status file returned from call.

   a. Abort if *status* != “Success”, display to the user the content of the **Message** text field.

   b. Save the *UID* locally to be used in future EDA operations.

10. Optionally set the local checkout status to false for future use.

11. Read EDADesign file returned from call.

    a. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save locally to identify design components uniquely in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute **UID**.

    b. If **updateTcEdaUID** is true then ECAD tool connector needs to update TcEdaUID property for ECAD components in ECAD design using rdn components attribute information.

12. Reopen the design if it was closed.

Save Derived Data

This operation allows the user to save derived datasets. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Create an EDADesign XML file (see **EDADesign** file example) with **saveDerivedData** info.

3. This operation can be executed using two ways. For a single step operation perform step 4 and 8. For a two step operation follow steps 5 to 8 onwards.

4. Call **–saveDerivedData** API.

5. Call **–preSaveDerivedData2** API.

6. Check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

    b. Optionally check the **DerivedDataset** elements to ensure that the required files are generated.

    c. For each *VariantBomOption* element, optionally check the **DerivedDataset** sub-elements to ensure that the required files are generated.

7. Call **–saveDerivedData2** API.

8. Check status file returned from call.

    a. Abort if *status* != “Success”, display to the user the content of the **Message** text field.

    b. Get the returned UID and update the previously saved UID locally to be used in future EDA operations if the UID is not empty. The UID change only occurs if the user revised the design parent item. No affect if only other items (PWB, Variant, Derived item) are revised.

Open

This operation allows the user to open an existing design stored in Teamcenter. The connector code should perform the following steps to implement this operation:

1. Call **–open** API.
2. Check status file returned from call.
    a. Abort if *status* != “Success”, display the content of the Message text field to the user.
3. Read **EDADesign** file returned from call.
    a. There may be multiple *Dataset* elements. The *type* attribute should be used to determine whether each is the primary data type you are looking for, or whether this represents a related design that was opened along with the primary. Repeat the following steps for each *Dataset* element as required.
    b. The *Dataset* element *src* attribute contains the folder that the design was extracted to.
    c. The *Dataset* element *UID* attribute contains the UID of the dataset, save it for future use.
    d. The *Dataset* element *checkedOut* attribute contains the checkout status of the design. You may wish to save it for future use.
    e. The *Dataset* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the dataset attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.
    f. The *CCAVariant* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the variant attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.
    g. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute **UID**.
4. Open the appropriate design file in the ECAD tool. Some ECAD systems may require existing open designs to be closed before opening a different design.

Check-Out

This operation allows the user to checkout the design currently open in ECAD tool. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Optionally, validate that the design is not currently checked out.

3. You should close the current design since this operation may need to overwrite the design files (if there is a later version stored in Teamcenter). The overwrite may be blocked by the OS if the files are in use. Save a reference to this design in case there is no refresh.

4. Call **–checkOut** API.

5. Check status file returned from call.

    a. If *status* != **Success**, display the content of the Message text field to the user, then reopen the previously closed design and abort.

    b. Get the *refreshed* flag and UID (*primaryDsUID*) for the primary design.

    c. Get the *refreshed* flag, *UID*, and *datasetType* for each related design from the *RelatedDSRefreshStatus* element(s).

6. For each design whose *refreshed* flag is “true”, read the EDADesign file returned from call to find its related *Dataset* element.

    a. There may be multiple *Dataset* elements. The *type* attribute should be used to determine whether each is the primary data type you are looking for, or whether this represents a related design that was opened along with the primary. Repeat the following steps for each *Dataset* element as required.

    b. The *Dataset* element *src* attribute contains the folder that the design was extracted to.

    c. The *Dataset* element *UID* attribute contains the UID of the dataset, save it for future use.

    d. The *Dataset* element *checkedOut* attribute contains the checkout status of the design. You may wish to save it for future use.

    e. The *Dataset* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the dataset attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

f. The *CCAVariant* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the variant attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

g. If **updateComponentOccurrenceInfo** is **true** in configuration then the EDADesign file has the Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute “UID”.

7. If the primary designs *refreshed* flag is **false** then simply reopen the previously closed design.

8. Optionally, set the local checkout status to true and set the design files to writable.

Cancel Check-Out

This operation removes the checkout lock of a dataset, and reopens the original version of the dataset. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Optionally, validate that the design is currently checked out.

3. You should close the current design since this operation will overwrite the design files. The overwrite may be blocked by the OS if the files are in use. Save a reference to this design in case there is no refresh.

4. Call **–cancelCheckout** API.

5. Check status file returned from call.

   a. If *status* != “Success”, display the content of the Message text field to the user, then reopen the previously closed design and abort.

   b. If the *doNotKeepFlag* attribute is “true”, close the current design and change the working directory so that the common-client can delete the design files and folders. You can probably skip the remaining tasks in this case.

6. Read **EDADesign** file returned from call.

   a. There may be multiple *Dataset* elements. The *type* attribute should be used to determine whether each is the primary data type you are looking for, or whether this represents a related design that was opened along with the primary. Repeat the following steps for each *Dataset* element as required. The Cancel Check-Out operation does not refresh related datasets, so there should only be a Dataset element for the primary design here.

b. The *Dataset* element *src* attribute contains the folder that the design was extracted to.

c. The *Dataset* element *UID* attribute contains the UID of the dataset, save it for future use.

d. The *Dataset* element *checkedOut* attribute contains the checkout status of the design. You may want to save it for future use.

e. The *Dataset* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the dataset attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

f. The *CCAVariant* element *attr*

subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the variant attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

g. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute **UID**.

7. Open the appropriate design file in the ECAD tool.

Refresh

This operation refreshes the current design with the latest version of the dataset. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Optionally, validate that the design is not currently checked out.

3. You should close the current design since this operation will overwrite the design files. The overwrite may be blocked by the OS if the files are in use. Save a reference to this design in case there is no refresh.

4. Call **–refresh** API.

5. Check status file returned from call.

a. If *status* != **Success** , display the content of the Message text field to the user, then reopen the previously closed design and abort.

b. Get the *refreshed* flag and UID (*primaryDsUID*) for the primary design.

c. Get the *refreshed* flag, *UID*, and *datasetType* for each related design from the *RelatedDSRefreshStatus* element(s).

6. For each design whose *refreshed* flag is **true**, read the EDADesign file returned from call to find its related *Dataset* element.

   a. There may be multiple *Dataset* elements. The *type* attribute should be used to determine whether each is the primary data type you are looking for, or whether this represents a related design that was opened along with the primary. Repeat the following steps for each *Dataset* element as required.

   b. The *Dataset* element *src* attribute contains the folder that the design was extracted to.

   c. The *Dataset* element *UID* attribute contains the UID of the dataset, save it for future use.

   d. The *Dataset* element *checkedOut* attribute contains the checkout status of the design. You may wish to save it for future use.

   e. The *Dataset* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the dataset attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

   f. The *CCAVariant* element *attr* subelements will contain attribute name value pairs that are mapped via Teamcenter server CAD Attribute Mapping Definitions. This is the reverse of the variant attribute mapping operation performed during save operations. Only attributes whose Teamcenter CAD Attribute Mapping Definition specifies Teamcenter as the “master” are provided.

   g. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute “UID”.

7. If the primary designs *refreshed* flag is “false” then simply reopen the previously closed design.

Purge Working Files

This operation removes referenced designs from EDA cache and staging directory. The connector code should perform the following steps to implement this operation:

1. If current design is referenced, close it. The operation will attempt to delete it and it’s empty directory, so you may want to change directory if possible so that the OS doesn’t prevent the deletion.

2. Call **–purgeCache** API.

3. Optionally check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

Item Info

This operation displays the information about the current design. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Call **–showItemInfo** API.

3. Optionally check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

Design Info

This operation displays information about the current design. The connector code should perform the following steps to implement this operation:

1. Validate whether you have a UID for the current design.

2. Create an EDADesign XML file (see *EDADesign file example*for an example) with UID info. UID may be empty in case design does not exist in Teamcenter. Note that no BOM information is included.

3. Call **–showDesignInfo** API.

4. Optionally check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

Logout

This operation logs the user out of Teamcenter. The connector code should perform the following steps to implement this operation:

1. Call either **–logout** or **–logoutNoConfirm** API.

2. Optionally check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

The **logoutNoConfirm** API should be called when exiting the ECAD application to release the Teamcenter license and the attached EDA common client.

BOM Compare

This operation allows the user to compare part numbers of the component instances of the current design with the most recently saved Teamcenter BOM to assist the user in identifying and reconciling differences. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.

2. Create an EDADesign XML file (see *EDADesign file example* for an example) with reconcilePartsSelect info. Note that no BOM information is included.

3. Call **–reconcilePartsSelect** API.

4. Check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user and abort the operation.

    b. If *BOMOption* indicates that the user requested comparison of the BOM, extract the component list from the design. This will be used to create CCA Component elements in a new EDADesign file (in step 5).

    c. The *VariantBomOption* element(s) contain the *variantName* and *BOMOption* for each variant specified by the user to be compared. Extract BOM as requested for each variant. This will be used to create CCA Variant elements in a new EDADesign file (in step 5).

5. Create an EDADesign XML file (see *EDADesign file example* for an example) that includes the requested BOM information.

6. Call **–reconcileParts** API.

7. Check status file returned from call.

    a. Abort if *status* != “Success”, display the content of the Message text field to the user.

8. Read EDADesign file returned from call.

    a. If **updateComponentOccurrenceInfo** is true in configuration then EDADesign file have Teamcenter occurrence identifier, UID for design components. Save a local copy to identify if the design components are unique in Teamcenter bill of material (BOM). This information is available in rdn section of component with attribute **UID**.

Issue Manager

This operation allows user to maintain issue reports for design at the EDA side and the user can create, modify an issue report, or check all issues related to current opened design. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.
2. Create an EDADesign XML file (see *EDADesign file example* for an example) with UID info. UID may be empty in case design does not exist in Teamcenter. Note that no BOM information is included.
3. Call initiate issue manager API.
4. Check status file returned from call.
    a. Abort if *status* != “Success”, display the content of the Message text field to the user.
5. Once user create an issue report into Teamcenter, the latest EDA design revision is automatically attached to issue report object.

Collaboration

This is a sub-menu heading, containing the following three collaboration menu options.

Initiate Collaboration Issue

This operation allows the user to maintain collaboration issues for the design across different domains (For example, the ECAD, MCAD, or CAE domains) during production. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.
2. Create an EDADesign XML file (see *EDADesign file example* for an example) with UID info. UID may be empty in case design does not exist in Teamcenter. Note that no BOM information is included.
3. Call initiate collaboration issue API.
4. Check status file returned from call.
    a. Abort if *status* != “Success”, display the content of the Message text field to the user.
5. Once user create an issue report into Teamcenter, the latest EDA design revision is automatically attached to collaboration issue.

Publish Collaboration File

This operation allows the user to upload collaboration files helping designers at different domains (for example, ECAD, MCAD, or CAE domains) to handle the issue, specially IDX file which carry collaboration information of the design. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.
2. Create an EDADesign XML file (see *EDADesign file example* for an example) with UID info. UID may be empty in case design does not exist in Teamcenter. Note that no BOM information is included.
3. Call publish collaboration file API.
4. Check status file returned from call.
    a. Abort if *status* != "Success", display the content of the Message text field to the user.

Download Collaboration File

This operation allows the user to download collaboration files from the collaboration issue, specially IDX file which carry collaboration information of the design across different domains. The connector code should perform the following steps to implement this operation:

1. Validate that you have a UID for this design.
2. Create an EDADesign XML file (see *EDADesign file example* for an example) with UID info. UID may be empty in case design does not exist in Teamcenter. Note that no BOM information is included.
3. Call download collaboration file API.
4. Check status file returned from call.
    a. Abort if *status* != "Success", display the content of the Message text field to the user.

Workflow

This is a sub-menu heading, containing the following three workflow menu options.

Design Workflow

The Design Workflow operation is used by the ECAD Designer to initiate a design workflow process for the current design.

ECAD Designers can initiate a design workflow with the **EDADesign.xml** file name that contains the current design information. The connector code performs the following steps to implement this operation:

1. Create an **EDADesign.xml** file containing design identification (Teamcenter UID assigned to primary dataset).

2. Create an empty EDAStatus file to receive operation status from the EDA common client.

3. Call the **–initiateDesignWorkflow** API to initiate the workflow (see **–initiateDesignWorkflow** API). An **Initiate a Workflow** wizard is executed.

   Workflow can be created from the available workflow templates.

4. Click the **Finish** button.

   The EDA common client creates a workflow in Teamcenter.

5. Check the operation status in the status file returned from the call.

   Abort if *status* != “Success”, display the content of the Message text field to the user.

Part Workflow

The Part Workflow operation is used by the ECAD Designer to initiate a part workflow process for the current design.

ECAD Designers can initiate a part workflow with the **EDADesign.xml** file name that contains the current design information. The connector code performs the following steps to implement this operation:

1. Create an **EDADesign.xml** file containing design identification (Teamcenter UID assigned to primary dataset), and BOM component information.

2. Create an empty EDAStatus file to receive operation status from the EDA common client.

3. Call the **–initiatePartWorkflow** API to initiate the workflow (see **–initiatePartWorkflow** API). An **Initiate a Workflow** wizard is executed.

   Workflow can be created from the available workflow templates.

4. Click the **Finish** button.

   The EDA common client creates a workflow in Teamcenter.

5. Check the operation status in the status file returned from the call.

Abort if *status* != “Success”, display the content of the Message text field to the user.

Workflow Status

The Workflow Status operation is used by the ECAD Designer to examine the process status for all processes initiated by the ECAD designer and processes associated with the PCA/PCABase of the current design.

1. Call **–showWorkflowStatus** API

2. Check status file returned from the call

   Abort if status != “Success”, display the content of the Message text field to the user.

Inbox

This operation allows the user to view his or her Teamcenter Inbox. The connector code should perform the following steps to implement this operation:

1. Call **–getMyWorklist** API.

2. Check status file returned from the call

   a. Abort if status != “Success”, display the content of the Message text field to the user.

6. About configuration file

Each new integration requires a unique configuration file to define the connector. This configuration file specifies information, including the application name, the family the application belongs to, the common client class to be used, and the Teamcenter dataset types to be used.

Name this file *applicationName***edadef.xml** (where applicationName is replaced with a unique value, see below), and save it in CLASSPATH for the common client to find it. It is typically saved in the TCEDAECAD_ROOT folder. This file is called system configuration file and basically determines the capability of the connector related to the named application.

Starting with Teamcenter EDA 2.4, an optional configuration file called user configuration file is allowed to alter some of the system configurations. Create and name this file *applicationName***usrdef.xml** and save it in the TCEDAECAD_ROOT folder.

The configuration file schema is defined in the **EDAClientDef** chapter of the **Schema Reference** document. The schema file may be found in %TCEDA_ROOT%\example\schema\EDAClientDef.xsd. Both system and user configuration files follow this same schema.

All attributes and elements of this schema are applicable to design connectors except for the following.

- The **LibraryDef** sub-element is **not** used in a design connector. It is only used in a library connector. See the Teamcenter EDA Teamcenter EDA Library Connector Integration — Reference for details on these connectors.
- The **EDAGatewayDef** sub-element is only used in a Teamcenter Gateway for EDA connector. See the Teamcenter Gateway for EDA Integration — Reference for details on these connectors.
- The **SaveOptionDefs** sub-element is reserved for future use.
- The **OperationData** sub-element is not used in a design connector. It is only used in a library connector. See the Teamcenter EDA Teamcenter EDA Library Connector Integration — Reference for details on these connectors.

All required attributes and elements must be specified in both configuration files if the user configuration is added. The user configuration should carry the same attribute and element specifications if they are not allowed to be overridden.

The following required or optional attributes and elements are allowed to be overridden by the user configuration file.

- family
- enableVariants Note 1
- supportsBaseBom Note 1

- allowVariantNameAsItemId Note 1
- enableOpenRelatedDatasets Note 1
- PrimaryDataType Note 2
- RelatedDataType Note 2
- PartitionDataType Note 2
- IssueAttachmentsDataType Note 2
- CallbackDefs Note 3
- RdnAttrMapDefs
- MigrationDef
- updateComponentOccurrenceInfo Note 1

Note 1: You cannot set this preference valve to **true** if it is not set to true by the system configuration. This value is ignored by the system if the corresponding value in the system configuration file is set to **false**.

Note 2: You cannot change the name and type, but you can specify your own **Root Level**, **Include**, and **Exclude**.

Note 3: Both system-configured and user-configured callbacks are used. The system-configured callback is executed first.

7. EDADesign file

The EDADesign file is used to pass information between the EDA common client and an EDA connector, or between the common client and a callback. The EDADesign file is UTF-8 character set encoded.

Most times the connector will recreate this file, and pass it to the common-client through the API call. However, in some cases the common-client will recreate this file and pass it back to the connector. In those cases the connector must parse the file to get design information from it.

The **EDADesign** file schema is defined in the **EDADesign** chapter of the **Schema Reference** document. The schema file is available in the *%TCEDA_ROOT%***\example\schema\EDADesign.xsd** folder.

A sample EDADesign file outline is shown below. Note that the EDADesign element contains a single CCA element, which contains all other EDA design elements.

```
<?xml version="1.0" encoding="UTF 8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
    <CCA name="pli002" itemId="adfdadfds" bom="false">
        <dataset type="schematic" UID="UINlqzw9hPJ4oD"
                 src="E:\cadenceTest\DataSets\brdtest" version="1" >
            <attr name="design_name" value="default.sch" />
        </dataset>
    </CCA>
</EDADesign>
```

EDADesign file example with CCAVariants. Note that the variant elements contain attr elements for Teamcenter CAD Attribute Mapping.

```
<?xml version="1.0" encoding="UTF 8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
    <CCA name="pli002" itemId="adfdadfds" bom="false"
     supportedBOMOptions=”FromPCB,FromSchematic”>
        <dataset type="schematic" UID="UINlqzw9hPJ4oD"
                 src="E:\cadenceTest\DataSets\brdtest" version="1" >
            <attr name="design_name" value="default.sch" />
        </dataset>
        <CCAVariant variantName="BGA" itemId="Var_11" revId="A"
bom="false"
                           supportedBOMOptions=”FromPCB,FromSchematic”>
            <attr name="Footprint" value="MCHP-SOIC-SN881"/>
        </CCAVariant>
        <CCAVariant> variantName="TSOP" itemId=" Var_12" revId="A"
bom="false"
                     supportedBOMOptions=”FromPCB,FromSchematic”>
            <attr name="Footprint" value="MCHP-SOIC-SN999"/>
```

```
            </CCAVariant>
        </CCA>
    </EDADesign>
```

All attributes and elements of this schema are applicable to all design connectors.

8. EDAStatus file

The EDAStatus file is used to return results and status from the EDA common client API to the EDA connector. The EDA Status file is UTF-8 character set encoded.

The EDAStatus file schema is defined in the **EDAStatus** chapter of the **Schema Reference** document. The schema file may be found in *%TCEDA_ROOT%***\example\schema\EDAStatus.xsd**.

The following status types should not be used in a design connector:

- **EDAExportLibraryStatus**
- **EDAImportLibraryStatus**
- **EDAGetPreferenceValuesStatus**
- **EDASetPreferenceValuesStatus**

The preSave, preSaveAs, preCheckIn, saveDerivedDataset APIs that contains the **DerivedDataset** elements, contains information of the selected derived datasets to be generated and saved. Each element contains the **name** and **pathname** attributes specified in the BMIDE derived data configuration models. The **name** is the configuration name and the **pathname** corresponds to the path specifications. For the **pathname**, the keywords in the path specifications are substituted with local paths, while the wildcards in the path specifications are left unchanged. Theses elements provide the optional info for the users to check/validate the derived datasets against the BMIDE models.

9. Common Client API

Overview of common client APIs

The EDA common client methods can be accessed through a *C* interface or a *command-line* interface.

The command-line interface is provided through a Windows batch script file (**edacli.bat**). This script launches a *Java Virtual Machine* (JVM) process, passing the required arguments. The same JVM is utilized for subsequent commands and is released when the ECAD Designer logs out.

- Application name must be passed in on each call.
- Teamcenter login session must be reestablished for each call. This is done inside the common-client through the use of a cookie, so it is not a cost to the connector.

The C interface is provided through a Windows library (**edabase.dll**). This interface launches the java process one time, subsequent method calls then connect to the same java process. The C interface methods return an int status:

- 0 – operation ended successfully
- 1 – operation failed
- –1 – operation was cancelled

All arguments to both interface types are input only. Note that, while the argument value is only input, it may represent a filename which the common-client will create (thus a pseudo-output).

Many of the interface methods use the same arguments, so they will be documented here rather than repeating these descriptions multiple times.

- **applicationName** – The application name representing the ECAD tool in use. This value specifies which configuration file to use.
- **EDADesignFileName** – An absolute pathname to an EDADesign file containing information to be passed in from the connector to the common-client.
- **EDAStatusFileName** – An absolute path for an EDAStatus file to be created by the common-client to return status to the connector.
- **DatasetUID** – The Teamcenter UID of the design dataset of the current design

setApplicationName

This method should be called to set the application name. This is only available through the C API, since the command line API passes in the application name in each call. This must be called before any other API calls.

C API

```
void EDA_setApplicationName(
 const char *applicationName );
```

preSaveAs2

This API should be called before saveAs2. It displays the Save As dialog, then returns the user input such as itemId, revision, dataset name, and so on.

C API

```
int EDA_preSaveAs2(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –preSaveAs2
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" bom="false" supportedBOMOptions=”FromSchematic”>
 <dataset
 type="schematic"
 src="E:\cadenceTest\DataSets\brdtest"
 version="1">
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPreSaveAsStatus
 BOMOption=”WithBOM”
 itemId=”adfdadfds”
 itemName=”pli002”>
 viewableOption=”WithViewable”
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

saveAs2

This API should be called to save a new design or save an existing design as copy.

C API

```
int EDA_saveAs2(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –saveAs2
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds” bom="true">
 <dataset type="schematic"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 <dataset type="schematicViewable"
 src="C:\Temp\Cadence\viewable\" checkedOut="false" />
 <component name="20L10" itemId="20L10" revId="–" quantity="1">
 <rdn name="U2" />
 </component>
 <component name="30L10" itemId="30L10" revId="–" quantity="1">
```

```
 <rdn name="U3" />
 </component>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDASaveAsStatus
 UID="0123456789"
 checkIn="true"
 status="Success"
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

preRevise

This API must be called before revise. It displays the Revise dialog, then returns the user input such as itemId, revision, and dataset name.

C API

```
int EDA_preRevise(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli -preRevise
 -application applicationName
 -edaDesign EDADesignFileName
 -status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" bom="false"
supportedBOMOptions="FromPCB,FromSchematic">
 <dataset
 type="schematic"
 src="E:\cadenceTest\DataSets\brdtest"
 version="1">
 <attr name="design_name" value="default.sch" />
```

```
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPreSaveAsStatus
 BOMOption=”WithBOM”
 itemId=”adfdadfds”
 itemName=”pli002”>
 viewableOption=”WithViewable”
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

revise

This API should be called to revise a design.

C API

```
int EDA_revise(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –revise
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds” bom="true">
 <dataset type="schematic"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 <dataset type="schematicViewable"
 src="C:\Temp\Cadence\viewable\" checkedOut="false" />
```

```
 <component name="20L10" itemId="20L10" revId="–" quantity="1">
 <rdn name="U2" />
 </component>
 <component name="30L10" itemId="30L10" revId="–" quantity="1">
 <rdn name="U3" />
 </component>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
 <EDASaveAsStatus
     UID=”0123456789”
     checkIn=”true”
     status=”Success”
     xmlns="http://www.ugs.com/tc/EDAStatus" />
 saveas
```

preSave

This API should be called before save. It will display the BOM option dialog, and returns the BOM option user selects.

C API

```
int EDA_preSave(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –preSave
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF–8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false"
 supportedBOMOptions=”FromPCB,FromSchematic”>
```

```
 <dataset type="schematic" UID="UINlqzw9hPJ4oD"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPreSaveStatus
 BOMOption=”WithoutBOM”
 viewableOption="WithViewable"
 status=”Success”
xmlns="http://www.ugs.com/tc/EDAStatus" />
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPreSaveStatus
 DesignTcStatus=”tc_unknown”
 message=”The design has not been saved to Teamcenter. Use save-as to
continue.”
 status=”Error”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

save

This API should be called to save a new version of an existing design, user will keep the checkout lock after this call.

C API

```
int EDA_save(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –save
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false" >
 <dataset type="schematic" UID=”UINlqzw9hPJ4oD”
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataSet>
 <dataset type="schematicViewable"
 src="C:\Temp\Cadence\viewable\" checkedOut="false" />
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDASaveStatus
 UID=”UINlqzw9hPJ4oD”
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

preCheckIn

This API should be called before checkin. It will display the Check-In dialog, and returns the BOM option the user selects, and whether to keep the local design.

C API

```
int EDA_preCheckin(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –preCheckIn
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false">
 <dataset type="schematic" UID="UINlqzw9hPJ4oD"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPreCheckinStatus
 BOMOption=”WithoutBOM”
 viewableOption="WithViewable"
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

checkIn

C API

```
C API
int EDA_checkIn(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –checkIn
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false">
```

```
 <dataset type="schematic" UID=”UINlqzw9hPJ4oD”
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDACheckInStatus
 UID=”UINlqzw9hPJ4oD”
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

saveDerivedData

This API should be called to save a new derived dataset.

C API

```
int EDA_saveDerivedData(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –saveDerivedData
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false">
 <dataset type="schematic" UID="UINlqzw9hPJ4oD"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDASaveDerivedDataStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

open

This API should be called to open a dataset. Note that the EDADesignFileName specifies the filename for a file that the common-client will create – the connector does not create this file before calling this method.

C API

```
int EDA_openDataSet(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –open
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA itemId="adfdadfds" name="pli002" revId="A" >
 <dataset UID="UINlqzw9hPJ4oD" checkedOut="false"
 src="E:\StagingDir\cadence\latest\adfdadfds\sch"
 type="schematic" version=”3” />
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAOpenDataSetStatus
```

```
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

checkOut

This API should be called to checkout a dataset, if the dataset is not the latest version, it will be refreshed automatically, and CAD tool should reopen the refreshed design. Note that the **EDADesignFileName** specifies the filename for a file that the common-client will create if the design is being refreshed – the connector does not create this file before calling this method.

C API

```
int EDA_checkOut(
 const char *DatasetUID,
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –checkout
 –application applicationName
 –UID DatasetUID
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDACheckOutStatus
 refreshed=”false”>
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

cancelCheckOut

This API should be called to remove the checkout lock of a dataset, the original version of the dataset will be downloaded, the CAD tool should reopened the original dataset. Note that the EDADesignFileName specifies the filename for a file that the common-client will create – the connector does not create this file before calling this method.

C API

```
int EDA_unCheckOut(
 const char *DatasetUID,
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –cancelCheckOut
 –application applicationName
 –UID DatasetUID
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDACancelCheckoutStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
EDADesign file example
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA itemId="adfdadfds" name="pli002" revId="A" >
 <dataset UID="UINlqzw9hPJ4oD" checkedOut=”false”
 src="E:\StagingDir\cadence\latest\adfdadfds\sch"
 type="schematic" version="5" />
 </CCA>
</EDADesign>
```

refresh

This API should be called to refresh the current design, only the referenced non-latest dataset will be refreshed. Note that the EDADesignFileName specifies the filename for a file that the common-client will create if the design is being refreshed – the connector does not create this file before calling this method.

C API

```
int EDA_refresh(
 const char *DatasetUID,
```

```
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –refresh
 –application applicationName
 –UID DatasetUID
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA itemId="adfdadfds" name="pli002" revId="A" >
 <dataset UID=”E8Al8YyDhPJ4oD” checkedOut=”false”
 src="E:\StagingDir\cadence\latest\adfdadfds\sch"
 type="schematic" version=”4” />
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDARefreshStatus
 primaryDsUID="E8Al8YyDhPJ4oD"
 refreshed=”true”>
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

purgeCache

This API should be called to purge the cache, all referenced design will be removed from cache and staging directory.

C API

```
int EDA_purgeCache(
 const char *EDAStatusFileName );
```

Command line API

```
edacli –purgeCache
 –application applicationName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAPurgeCacheStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

itemInfo

This API should be called to get the information of a design. This method may still be used to get the design information returned to the connector via an EDADesign file. However, the new showDesignInfo method should be used to display the Design Info to the user, as that is now done via a common-client dialog.

C API

```
int EDA_itemInfo(
 const char *DatasetUID,
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –itemInfo
 –application applicationName
 –UID DatasetUID
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

showDesignInfo

This API should be called to show the information about a design in Teamcenter. If there is no current design open which has been saved to Teamcenter, then the users Home will be shown.

C API

```
int EDA_showDesignInfo(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –showDesignInfo
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false" >
 <dataset type="schematic" UID=”UINlqzw9hPJ4oD”
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataSet>
 <dataset type="schematicViewable"
 src="C:\Temp\Cadence\viewable\" checkedOut="false" />
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAGetItemInfoStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

getStagingDir

This API should be called to query the absolute path of the staging directory

C API

```
char * EDA_getStagingDir( );
```

```
return: Absolute path of staging directory.
char * EDA_getStagingDir(
 const char *EDAStatusFileName );
```

Command line API

```
edacli –getStagingDir
 –application applicationName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDAGetStagingDirStatus
 stagingDir=”E:\StagingDir\cadence”>
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

setPreferencesByFile

This API should be called to set preferences from a supplied file. The preferences are set from the contents of the supplied preference file. Unique arguments to this method are:

preferenceFileName – The absolute path of the preference file to be read.

C API

```
int EDA_setPreferenceValues(
 const char *preferenceFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –setPreferencesByFile
 –application applicationName
 –preferences preferenceFileName
 –status EDAStatusFileName
```

Preference file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes"?>
```

```
<EDAPreferences xmlns="http://tempuri.org/XMLSchema.xsd">
 <EDAPref Type="string" name="Preference_A" scope="user"
 value="ShyamTest2"/>
 <EDAPref Type="int" name="Preference_B" scope="user" value="0"/>
 <EDAPref Type="int" name="Preference_C" scope="user" value="1"/>
 <EDAPref Type="string" name="PreferenceD" scope="user" value="19"/>
 <EDAPref Type="string" name="PreferenceE" scope="user"
 value="lov_c"/>
</EDAPreferences>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDASetPreferenceValuesStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

getPreferences

This API should be called to get the current preferences from Teamcenter. The preferences are written into a new file as named by the preferenceFileName argument. Unique arguments to this method are:

- preferenceFileName – The absolute path of a preference file. The connector should create a preference file to be passed in containing the preferences desired. The common-client will overwrite this file with values filled in.

C API

```
int EDA_getPreferenceValues(
 const char *preferenceFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –getPreferences
 –application applicationName
 –preferences preferenceFileName
 –status EDAStatusFileName
```

Preference file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDAPreferences xmlns="http://tempuri.org/XMLSchema.xsd">
```

```
 <EDAPref Type="string" name="Preference_A" scope="user" value=""/>
 <EDAPref Type="int" name="Preference_B" scope="user" value=""/>
 <EDAPref Type="int" name="Preference_C" scope="user" value=""/>
 <EDAPref Type="string" name="PreferenceD" scope="user" value=""/>
 <EDAPref Type="string" name="PreferenceE" scope="user" value=""/>
</EDAPreferences>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAGetPreferenceValuesStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

logout

This API should be called to logout of Teamcenter, after confirmation from the user via a UI dialog.

C API

```
int EDA_logout(
 const char *EDAStatusFileName );
```

Command line API

```
edacli –logout
 –application applicationName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDALogoutStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

logoutNoConform

This API should be called to logout of Teamcenter without any confirmation from the user.

C API

None.

Command line API

```
edacli –logoutNoConfirm
 –application applicationName
 –status EDAStatusFileName
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDALogoutStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

reconcilePartsSelect

This API is used to select the BOMs available for a design, for comparison, and should be called before reconcileParts. It will display the BOM Compare dialog and return the user selected options needed for the reconcileParts call.

C API

```
int EDA_reconcilePartsSelect(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –reconcilePartsSelect
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF–8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds" bom="false">
```

```
 <dataset type="schematic" UID="UINlqzw9hPJ4oD"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAReconcilePartsSelectStatus
 BOMOption=”WithBOM”
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

reconcileParts

This API is used to provide the design and BOM details for BOM comparison that were requested and returned in the reconcilePartsSelect status file.

C API

```
int EDA_reconcileParts(
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –reconcileParts
 –application applicationName
 –edaDesign EDADesignFileName
 –status EDAStatusFileName
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds” bom="true">
 <dataset type="schematic"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 <dataset type="schematicViewable"
```

```
 src="C:\Temp\Cadence\viewable\" checkedOut="false" />
 <component name="20L10" itemId="20L10" quantity="1">
 <rdn name="U2" />
 </component>
 <component name="30L10" itemId="30L10" quantity="2">
 <rdn name="U3" />
 <rdn name="U4" />
 </component>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAReconcilePartsStatus
 status=”Success”
xmlns="http://www.ugs.com/tc/EDAStatus" />
```

initiateIssueReport

This operation opens issue manager dialog for user to perform related operations of issue report. The latest design revision is automatically attached under issue report based on the information provided in **edaDesign** xml file.

Command line API

```
edacli
–initiateIssueReport
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –initiateIssueReport –application mentorExpLib –edaDesign
C:\temp\edadesign.xml –status C:\temp\status.xml
EDADesign file example
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
```

```
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
xmlns="http://www.ugs.com/tc/EDAStatus" />
```

initiateCollaborationIssue

This operation opens issue manager dialog for user to create a collaboration issue into Teamcenter. The latest design revision is automatically attached under issue report based on the information provided in edaDesign xml file.

Command line API

```
edacli
–initiateCollaborationIssue
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –initiateCollaborationIssue –application mentorExpLib –
edaDesign C:\temp\edadesign.xml –status C:\temp\status.xml
EDADesign file example
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
< EDALibStatus
```

```
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

publishCollaborationFiles

This operation opens Publish Collaboration File dialog for user to upload files from local disk to Teamcenter. EDA design item revision is displayed by default based on information in edaDesign xml file.

Command line API

```
edacli
–publishCollaborationFiles
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –publishCollaborationFiles –application mentorExpLib –
edaDesign C:\temp\edadesign.xml –status C:\temp\status.xml
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
xmlns="http://www.ugs.com/tc/EDAStatus" />
```

downloadCollaborationFiles

This operation opens Download Collaboration File dialog for user to download issue attachments from Teamcenter to local disk. EDA design item revision is displayed by default based on information in edaDesign xml file.

Command line API

```
edacli
–downloadCollaborationFiles
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –downloadCollaborationFiles –application mentorExpLib –
edaDesign C:\temp\edadesign.xml –status C:\temp\status.xml
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

initiateDesignWorkflow

This operation is used by the ECAD Designer to initiate a design workflow process for the design specified in the EDADesign XML file.

Command line API

```
edacli
–initiateDesignWorkflow
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –initiateDesignWorkflow –application mentorExpLib –edaDesign
C:\temp\edadesign.xml –status C:\temp\status.xml
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

initiatePartWorkflow

This operation is used by the ECAD Designer to initiate a part workflow process for the design specified in the EDADesign XML file.

Command line API

None.

Command line example

```
edacli.bat –initiatePartWorkflow –application mentorExpLib –edaDesign
C:\temp\edadesign.xml –status C:\temp\status.xml
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_19236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
 <component name="20L10" itemId="20L10" revId="–" quantity="1">
 <rdn name="U2" />
 </component>
 <component name="30L10" itemId="30L10" revId="–" quantity="1">
 <rdn name="U3" />
 </component>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

checkPartWorkflowStatus

This operation is used by ECAD Designer to examine the process status for all processes they initiated and processes associated with the PCA/PCABase of the current design.

Command line API

```
edacli
–checkPartWorkflowStatus
 –application applicationName
 –edaDesign EDA Design file
–status status file
```

Command line example

```
edacli.bat –checkPartWorkflowStatus –application mentorExpLib –edaDesign
C:\temp\edadesign.xml –status C:\temp\status.xml
```

EDADesign file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
<CCA name="DxExp" itemId="000419" bom="false">
 <dataset
src="D:\ecad\EDAStaging83\gls\user3_-2082819236\expedition\latest\DxExp1"
 type="combined" UID="glN9Zk_zIRKtPB"></dataset>
</CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
< EDALibStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

getMyWorklist

This API opens a dialog displaying the users Teamcenter Inbox.

C API

```
int EDA_getMyWorkList(
 const char *EDAStatusFileName );
```

Command line API

```
edacli –getMyWorklist
 –application applicationName
 -status EDA Status File
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

progress

This API is called to display a progress dialog from the common-client. Unique arguments to this method are:

- text – Text to display in the progress dialog
- minSeconds – The minimum number of seconds to display the dialog.
- maxSeconds – The maximum number of seconds to display the dialog.
- waitforFileName – The absolute path of a file to watch. Terminate the progress dialog when this file no longer exists.

C API

None.

Command line API

```
edacli –progress text
 –application applicationName
 –min minSeconds
 –max maxSeconds
 –waitfor waitforFileName
```

downloadBOM

Retrieve the BOM for a design, as specified by the datasetUID. The EDADesignFileName is not an input file, but a file to be created by the operation. This file will contain the BOM.

C API

```
int EDA_downloadBOM(
 const char *datasetUID,
```

```
 const char *EDADesignFileName,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –downloadBOM
 –application applicationName
 –uid dataset UID
 -edaDesign EDA Design File
 -status EDA Satus File
```

Design file example

```
<?xml version="1.0" encoding="UTF-8" ?>
<EDADesign xmlns="http://www.plmxml.org/Schemas/tceda">
 <CCA name="pli002" itemId="adfdadfds” bom="true">
 <dataset type="schematic"
 src="E:\cadenceTest\DataSets\brdtest" version="1" >
 <attr name="design_name" value="default.sch" />
 </dataset>
 <component name="20L10" itemId="20L10" revId="-" quantity="1">
 <rdn name="U2" />
 </component>
 <component name="30L10" itemId="30L10" revId="-" quantity="1">
 <rdn name="U3" />
 </component>
 </CCA>
</EDADesign>
```

Status file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<EDAStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

copyDesignFolder

This method copies ECAD design files from source location to target location utilizing the application configuration includes/excludes for the specified data type.

C API

```
int EDA_copyDesignFolder (
 const char *originalFolder,
 const char *newFolder,
 const char *dataType,
 const char *EDAStatusFileName );
```

Command line API

```
edacli –copyDesignFolder
 –application applicationName
 –source Source folder location
 -target Target folder location
 -dataType Data type of the design, i.e. schematic, pcb, or simulation
 -status EDA Satus File
```

Status file example

```
<?xml version="1.0" encoding="UTF–8" standalone="yes" ?>
<EDAStatus
 status=”Success”
 xmlns="http://www.ugs.com/tc/EDAStatus" />
```

custom <operationName>

This method calls the callback defined in the connector configuration file. The behavior of this API depends on the defined callback the user provides.

C API

```
Not supported
```

Command line API

```
edacli –custom      <operationName>
          -application <applicationName>
          [-edaDesign  <edaDesignPathname>]
          [-status     <statusPathname>]
          [...]
```

The **custom** option and **operationName** argument are required. They, in the form of custom_<operationName>, specify the new operation, that is, which callback is to be invoked.

The **application** option and **applicationName** argument are required. These have the same usage as existing API's.

The **edaDesign** and **status** options are optional. If specified the appropriate argument is required. These have the same usage as existing API's.

Any optional additional arguments are simply passed along to the callback executed by this API. No dash (-) is allowed as the beginning letter for these additional arguments. The dash is reserved for the names of API options.

This new API performs the following tasks.

- Reconnect to an existing login session if one is available, or login to Teamcenter and start a new session.
- Execute the custom callback defined in the <applicationName>_edadef.xml file.
- Write the status file if the file path is specified.
- Return

This API is useless unless a custom callback is defined for it. Use the same way as for other supported callbacks to configure your custom callbacks. You can defined different callbacks with different operationName to provide multiple operational APIs.

For java callbacks, the method(s) with the following fixed arguments must be implemented.

```
public void methodNameX(java.lang.String operationName,
           java.lang.String applicationName,
           java.lang.String edaDesignPathName,
           java.lang.String statusPathName,
           java.lang.String[] extraArgs ){}
```

For Script callbacks, all arguments from the edacli call are passed to it sequentially. If the optional argument(s) are not specified, the temporary file path(s) will be filled to keep the argument position unchanged. The %1 corresponds to the operationName. The script callback should check the input arguments to assist developing and debugging.

Examples

These examples assume you have modified the connector configuration file (for example, *xpeditionPcb_edadef.xml*) with the following callback section.

```
<CallbackDefs>
    <callback operation="custom_compare" type="java"
              command="com.xplm.CustomClass:compare"/>
    <callback operation="custom_view" type="java"
              command="com.xplm.CustomClass:view"/>
    <callback operation="custom_check" type="script"
              command="checkTest.bat"/
   </CallbackDefs>
```

Now, you need to implement the following callbacks

The java callbacks:

```
Package com.xplm;
Import ...
Public class CustomClass
{
    public void compare( String   operationName,
                         String   applicationName,
                         String   edaDesignPathname,
                         String   statusPathname,
                         String[] extraArgs )
    {
        // Show that you can ignore the extraArgs
        System.out.println( “The operation name is:     “ +
operationName);
        System.out.println( “The application name is:   “ +
applicationName );
        System.out.println( “The edaDesign pathname is: “ +
edaDesignPathname);
        System.out.println( “The status pathname is:    “ +
statusPathname );
    }

    public void view( String   operationName,
                      String   applicationName,
                      String   edaDesignPathname,
                      String   statusPathname,
                      String[] extraArgs )
    {
        // Show that two optional arguments are not used
        System.out.println( “The operation name is:     “ +
operationName);
        System.out.println( “The application name is:   “ +
applicationName );
        // Show that extra arguments are used
        if( extraArgs != null)
```

```
        {
            for( String arg : extraArgs )
            {
                System.out.println( “An extra argument is:    “ + arg );
            }
        }
    }
    }
```

The script call back: checkTest.bat

```
echo off
setlocal
if '%1' == '' goto :EOF
echo Show first two arguments passed to this command
echo %1, %2
echo Show all arguments passed to this command
echo %*

endlocal
```

With the above setups, you may do the following command-line API calls

```
edacli -custom compare -application xpeditionPcb -edaDesign
c:\edaDesign.xml -status c:\status.xml extraArg1 extraaArg2 extraArg3
extraArg4
Note: all axtra arguments will be ignored

edacli -custom compare -application xpeditionPcb -edaDesign
c:\edaDesign.xml -status c:\status.xml
Note: just call with all required arguments

edacli -custom view -application xpeditionPcb extraArg1 extraArg2
extraArg3
Note: all required and axtra arguments are used

edacli -custom view -application xpeditionPcb -edaDesign
c:\edaDesign.xml -status c:\status.xml extraArg1 extraaArg2 extraArg3
extraArg4
Note: all required and axtra arguments are used.  Two optional arguments
are ignored

edacli -custom checkTest -application xpeditionPcb -edaDesign
c:\edaDesign.xml -status c:\status.xml extraArg1 extraArg2 extraArg3
Note: checkTest xpeditionPcb c:\edaDesign.xml c:\status.xml extraArg1
extraArg2     extraArg3 are passed as the inputs starting from %1
```

```
edacli -custom checkTest -application xpeditionPcb extraArg1 extraArg2
Note: checkTest xpeditionPcb edaDesignXXX.xml statusXXX.xml extraArg1
extraArg2 extraArg3 are passed as the inputs starting from %1
```