TEAMCENTER

Teamcenter EDA Library 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

1. About this document

This document provides supplementary information for the developers to create new or modify existing Teamcenter EDA (Electronic Design Automation) library 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:

- Teamcenter Server configuration
- ECAD tool configuration
- EDA client configuration

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

2. Supported integration types for library connectors

There are two methods of integrating within EDA. The traditional embedded integration enables an ECAD-tool centric integration, while the gateway integration enables library integration outside of the ECAD library tool.

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 Customizing Teamcenter EDA Gateway Integrations help.

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 export a library and to check out, save, check in, and synchronize parts from Teamcenter.

Embedded ECAD integrations pass data and status back and forth with the EDA common client, using an EDALib file and a status file. The EDALib file contains information about the ECAD library parts that you wish to save to or retrieve from Teamcenter. The EDA common client provides two sets of interfaces for integration.

- Command line API is used by ECAD tools that can only use scripts.
- JAVA API is used by ECAD tools that can invoke JAVA functions.

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

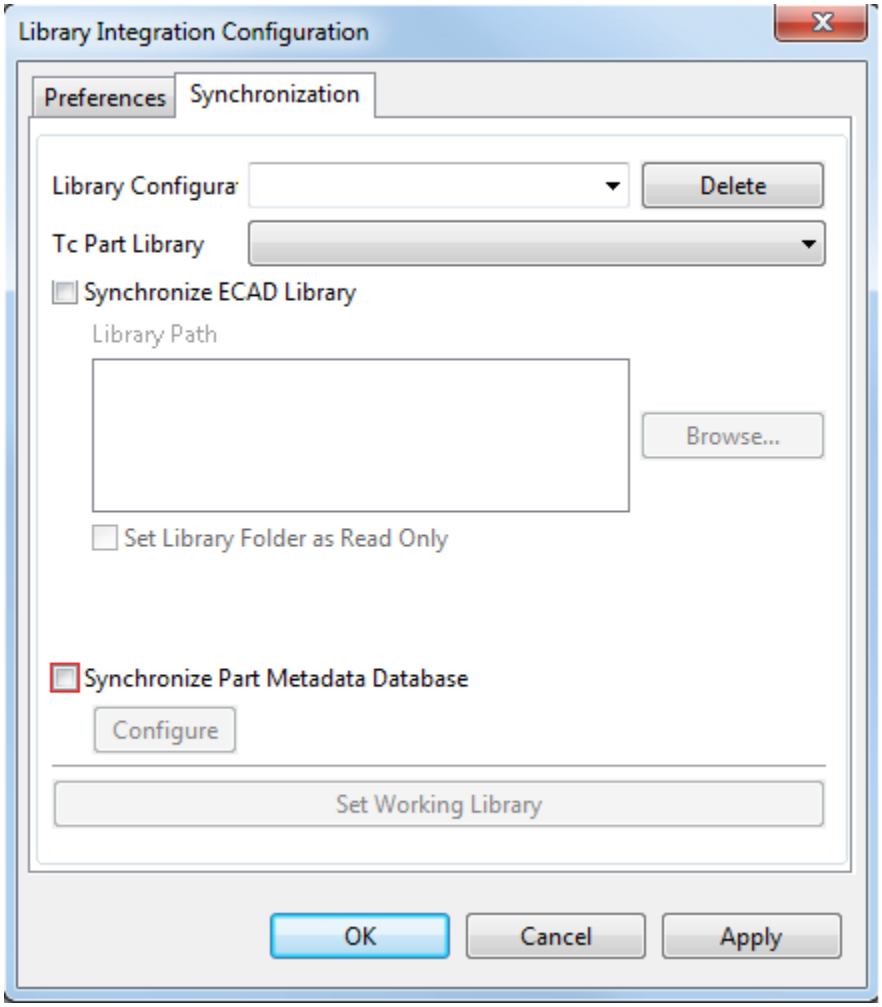
- Reading the client configuration file and performing appropriate action on the contents.
- Displaying user interface dialog boxes to gather user input, and providing Teamcenter navigation and search.
- Reading the EDALib file and calling the service-oriented architecture (SOA) to import data into Teamcenter.
- Calling the SOA to obtain data from Teamcenter and writing an EDALib file.
- Zipping and uploading ECAD library parts as well as its associated objects (symbol or footprint) to Teamcenter, downloading, and updating EDA library data based on the EDALib content from the Teamcenter side. This only applies if EDALib file contains any dataset object.

3. Supported library data types

The EDA library integration mainly supports two types of library data:

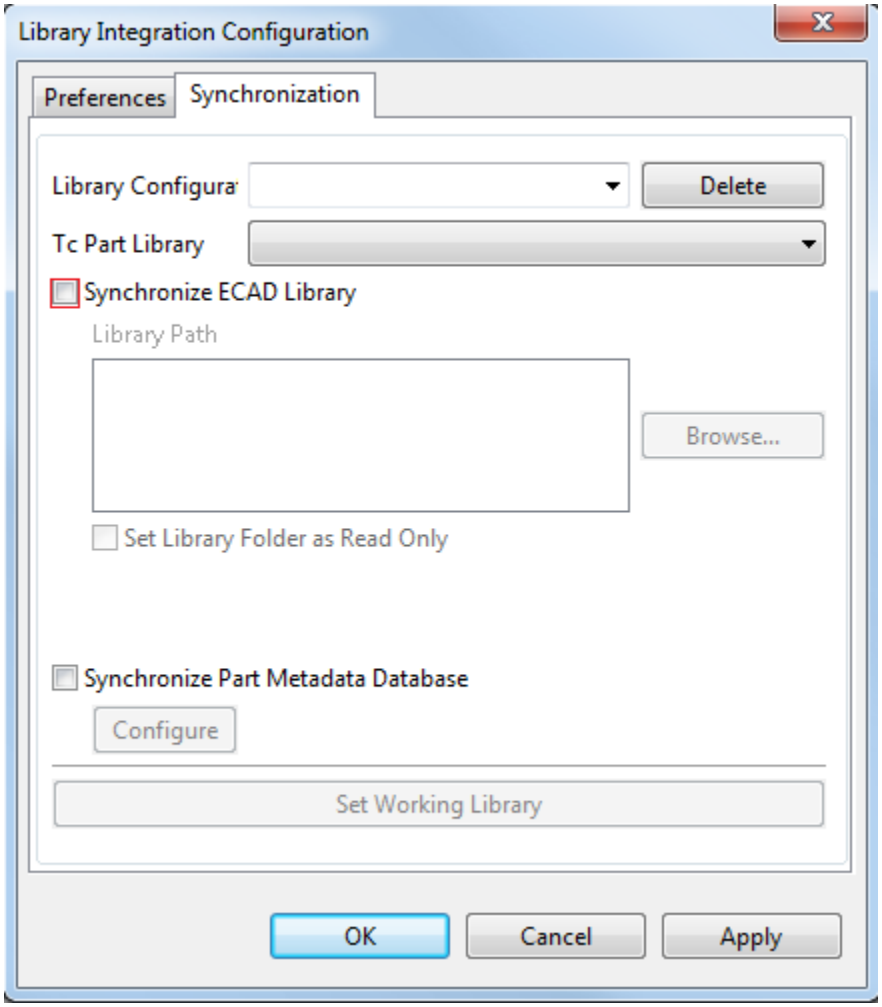
Part library

This library type manages library parts as well as the part, symbol, and footprint attributes without any information on the graphic entities with Teamcenter.



Shape library

This library type generally manages the library part, system, footprint, and part attributes along with information about the graphic entities for all objects with Teamcenter.



4. Create a new EDA library integration

High level steps to create a new library integration

In order to perform a new integration, you must complete the following tasks.

1. **Perform Teamcenter server configurations**

   For any new library integration, verify the part library management configurations required in the Teamcenter server.

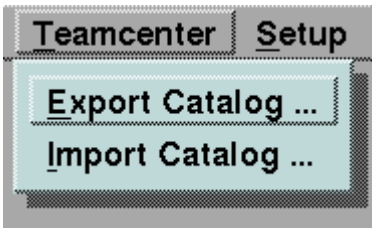
2. **Create Configuration files**

   Create configuration files for each ECAD tool you want to integrate. Refer to the Schema Reference help for the complete description of the configuration files.

3. **Create a code for embedded Teamcenter EDA library operations**

   Create a code to embed a menu or command buttons and the underlying code into the ECAD tools. The menu or dialog box should be configured as below:

   Each menu command must run the embedded code to perform the steps documented in the **Embedded Teamcenter EDA Library Operations** section.



   This code must read and write **EDALib.xml** files that are fully documented in the section EDALib. The code will also need to read but not write the **EDAStatus.xml** files, which are fully documented in the section Status file.

Teamcenter server configurations for library integration

Perform Teamcenter server configurations

You require following configurations on the Teamcenter server before you create a new library integration.

- **Map ECAD objects to Teamcenter objects**

- **Classification and attribute mappings**

Map ECAD objects to Teamcenter objects

This topic explains configurations to manage ECAD objects with Teamcenter objects. For details instructions to perform the following steps, refer Map ECAD library objects to Teamcenter objects topic in the *Xpedition and PADS Integration With Teamcenter* help.

Create Business Objects

Use Business Modeler IDE to create new business objects in Teamcenter that represent models in ECAD tools. You can use existing business objects such as EDAComPart, EDASymbol, EDAFootprint, and EDAPadstack, that are provided by the EDA library template.

Create dataset types

A primary dataset type is required only if your integration manages files for a shape library. You can use the existing dataset types provided by Teamcenter or create your own dataset type, using Business Modeler IDE.

Map ECAD objects with Teamcenter business objects

Modify the following preferences to map ECAD objects to Teamcenter business objects:

- **EDALIB_ItemTypesMapping**
- **EDALIB_GDETypesMapping**
- **EDALIB_DatasetTypesMapping**

Update ICS_classifiable_types

When a new library integration uses a classification object to maintain part attributes, update the **ICS_classifiable_types** preference to enable classification of the Teamcenter objects.

Configure manage rule

Update the **EDALib_ManagedObjects** preference with information about what relations and objects should be loaded or updated while downloading or updating data between ECAD and Teamcenter. An example is as follows:

- **Part Library**

```
EDALib_ManagedObjects  = Component;view;Symbol,Footprint;Write
                         Symbol;view;Symbol;Write
```

```
                                    Footprint;view;Footprint;Write
                                    Footprint;EDAPadstackRelation;Padstack;Write
```

When loading a component from Teamcenter to an ECAD tool, the EDA common client loads the component and its attributes. Based on this preference, all symbol and footprint objects are loaded in the component revision BOM view.

For each symbol object and footprint object the common client parses the revision bom view and loads all sub objects. The client also loads the padstack object that links to the footprint with the **EDAPadstackRelation** relation. The EDA common client then updates this information in the **EDALib.xml** file to the connector.

While updating the components from the ECAD tool into Teamcenter, the EDA common client updates all attributes. Based on the values set for this preference, the client adds or removes all symbol and footprint objects with information from the **EDALib.xml** file connector. For each symbol object, the client goes through the revision bom view and removes, updates, or adds sub symbol objects based on **EDALib.xml** content. For each footprint object, the client goes through the revision bom view and removes, updates, or adds sub footprint objects based on the **EDALib.xml** content, as well as removes, updates, or adds padstack object that links to footprint with **EDAPadstackRelation** relation.

- **Shape Library**

```
EDALib_ManagedObjects  = Component;view;Symbol,Footprint;Write
                         Component;IMAN_reference;cadenceLib;Write
                         Symbol;view;Symbol;Write
                         Symbol;IMAN_reference;cadenceLib;Write
                         Footprint;view;Footprint;Write
                         Footprint;IMAN_reference;cadenceLib;Write
                         Footprint;EDAPadstackRelation;Padstack;Write
```

The manage rule for shape library is very similar to part library integration. This rule extends values to maintain the primary dataset (for example, **cadenceLib** dataset) for shape library integration.

Classification and attribute mapping

This section explains how to set up an environment for classification and attribute mapping. For more information, refer the Set up classification mapping and Mapping ECAD part attributes to Teamcenter attributes topics in the Integrating Mentor Xpedition and PADS with Teamcenter help.

Classification Mapping

You can import and export classification hierarchy data using either the import and export features in the Classification Admin application or using command line utilities. Classification Admin supports both legacy XML files, as well as the industry standard PLM XML.

As a library administrator, you can create a custom classification schema using the Teamcenter Classification Admin application.

After you import or create the classification schema, you must set up Classification mapping. This is a one-time process and the classification mapping must be updated only if the classification schema is changed. Classification mapping converts the part category in the ECAD tool library to the class ID defined in Teamcenter Classification and the class ID in Teamcenter Classification to the part category in ECAD tool library.

Because the classification schema may contain a long list of classes, the **edalib_generate_classification_csv_maps** utility is provided to prepopulate the Classification mapping files.

Example:

Consider that the EDALib.xml file contains the following category information:

```
<item elemId="id18" name="comp01_1" libraryId="id17"
itemId="comp01_1"
type="Component" category="Electronic Parts,RESISTOR,FIXED,LINEAR"
childList="id19 id22 id25">
```

The EDA common client extracts the value of category in EDALib.xml file, searches the csv file for the mapped Teamcenter classification class id, and then creates the classification class and links it to the item revision.

To skip the classification process, do not specify a value.

Part attribute mapping

When attribute mapping is configured as follows:

```
{ Item type="EDAComPart"
Cost : Item.GRM(IMAN_master_form).object_desc /description="Cost"
Tolerance : ItemRevision.GRM(IMAN_master_form).object_desc /
description="Tolerance"
}
```

The attributes in the EDALib.xml file are saved in Teamcenter as follows:

```
<item elemId="id18" name="comp01_1" libraryId="id17" itemId="comp01_1"
type="Component" category="Electronic Parts,RESISTOR,FIXED,LINEAR"
childList="id19 id22 id25">
<attr name="Cost" value="10" type="String"/>
```

```
<attr name="Tolerance" value="34%" type="String"/>
 </item>
```

The value of the Cost attribute is saved in the description of the IMAN master form against **EDAComPart**, while the value of Tolerance is saved in the description of the IMAN revision master form against the **EDAComPart** revision. If attribute mapping is not configured, the EDA common client will ignore attributes during client operations.

Creating configuration files

Create configuration files

Create configuration files

Each new connector creates a configuration file to define the integration. 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 the configuration 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.

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

This section only covers the elements that are used in EDA library integration.

EDAClientDef

This is the main section of the schema, and it contains the following attributes and elements.

application

This is a required attribute that specifies the application name for this connector. This name must match that specified at the beginning of the file name (for example, for *<EDAClientDef application="mentor" .../>*, the filename must be **mentor_edadef.xml**). In addition, this is the name passed from the ECAD tool to the common client in the application argument.

EDA library integration currently recognizes the following applications and their application names out of the box.

Application Type	Application Name
Mention Expedition	mentorExpLib
Mentor Pads	mentorPadsLib
Mentor board station	mentorBSLib
Mentor Capital	mentorCapital

The EDA library integration supports custom application types in Organization. For details refer to **Customize application type** topic.

className

This is a required attribute and specifies the common client class to be used as the basis for this integration. You have the option of using the default class, or you can specify a subclass to be used here.

`com.teamcenter.edabase.lib.DefaultLibraryFactory`

This is the default class. This class has service and UI factory elements that are used by library integration.

family

This is a required attribute and specifies the family name for a set of configurations.

preserveSession

This is an optional attribute and specifies whether the login information should be preserved, using cookies. The default for this is **false**. Set it to **true** for integrations that make calls to the common client via the **edacli.bat** file. This is because such calls start a new process that would otherwise require the user to log on again.

licenseKey

This optional string attribute can contain a Teamcenter feature key that must be validated and checked out before allowing this configuration to be used. If this attribute is specified, then the given value will be validated and checked out via the Teamcenter license checking. In addition to this specified value, the library gateway feature key will also be validated and checked-out. The default is "", which will cause the internal mapping to be used.

PrimaryDataType

This element defines the dataset type to be used when storing native graphic entity information to Teamcenter. It contains the following elements and attributes.

datasetType

This is a required attribute that specifies the Teamcenter dataset type. This attribute is required for the shape library.

CallbackDefs

The <**CallbackDefs**> element contains callbacks that are used at various points in the EDA integration process. Using this, you can write custom code to perform tasks specific to this ECAD tool. Each <callback> element contains three required attributes: type, command, and operation.

For details on callback usage, refer to the documentation in the installed *eda\example* directory. For gateway specific callback help, refer to Teamcenter Gateway for EDA Integration — Reference doc.

LibraryDef

This section is used to define whether the library data type is Part Library or Shape Library, or one that is manually updated through imported or exported files.

SelectLibraryPathInImport

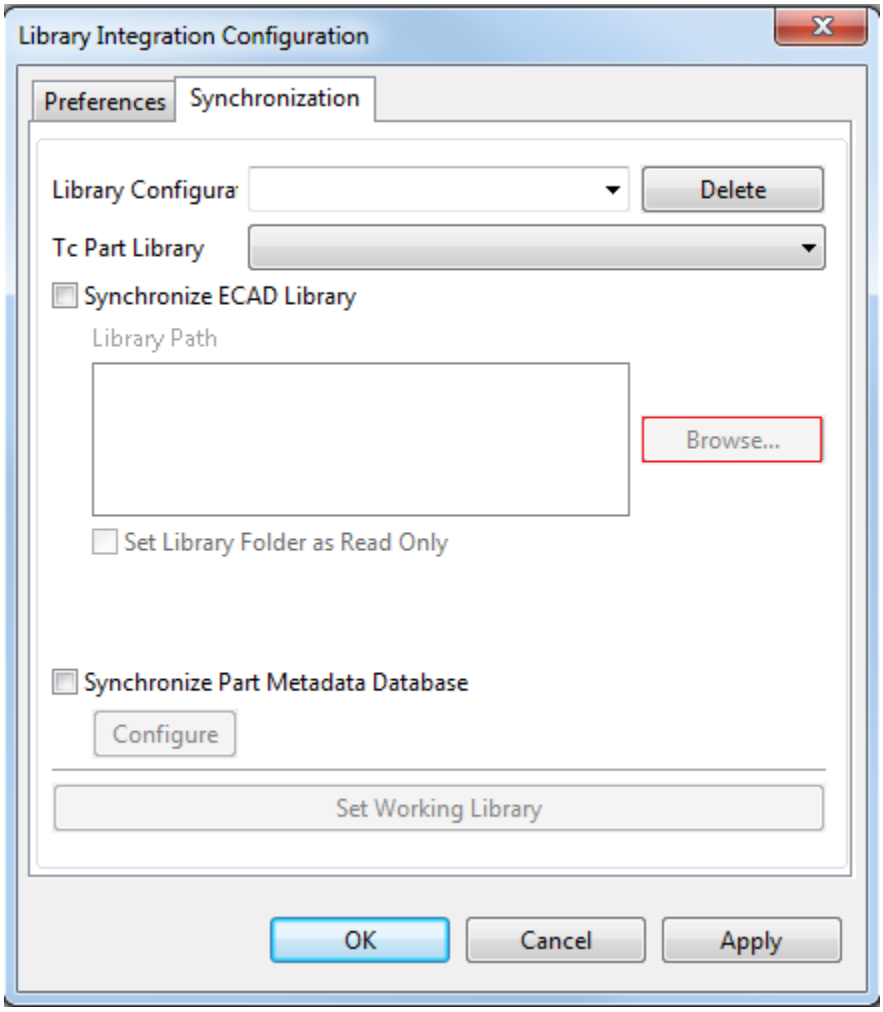
This attribute is obsolete in library integration.

SyncLibrary

Set this attribute to **true** for a shape library integration. In this case the **Synchronize ECAD Library** checkbox in **Teamcenter** > **Configure** > **Synchronization** tab is enabled, and the user clicks the **Browse...** button to define the shape library path. This button action is defined by the **Extension** attribute.

Extension

This is an optional attribute and only applies to shape library integrations. You can create the configuration from the EDA gateway **Teamcenter** > **Configure** > **Synchronization** tab, and click **Browse...** to view the **Extension** attribute.



- Omit the attribute if you are customizing **Library Path**, but have already defined the configureLibraryPath callback. The **Browse** button invokes the callback.

- When no configureLibraryPath callback is defined but **Library Path** is an extension file, you must set this value, for example, Extension="*.lmc." The **Browse** button filters only the valid files in the dialog.

- Click the **Browse** button to specify a folder path when **configureLibraryPath** and **Extension** are not set.

SyncDatabase

Set this attribute to **true** for a part library integration. In this case the **Synchronize Part Metadata Database** check box in **Teamcenter** > **Configure** > **Synchronization** tab is enabled. The new integration implements the **configureMetadataDB** callback to for Part Library configuration.

SyncFile

Set this attribute to **true** for a file based library integration.

MultipleRevision

Set this attribute to **true** for a multiple revision support in Capital Library Integration Def file. When the **MultipleRevision** attribute is set to true, **ExportLibrary**, **SyncLibrary**, and **LoadLibrary** operations can work with multiple revisions of the same item. That is, when an item info passed in the edalib.xml file through the connectors have revision info, the export, sync and load operations can work on specific revision of that item.

OperationDataType

This section is used to support operation extension points. For example, when you save a new part in Teamcenter through the saveLibrary() API, the part does not link with any external application in Teamcenter. However, in some cases, if you want to save a new part into Teamcenter and also link it with Teamcenter external applications, add the following code to the **edadef** file.

```
<OperationData operationName="SaveLibraryOperation">
        <Attr name="linkToTCLib" value="true"/>
    </OperationData>
```

Embedded Teamcenter EDA Library operations

About the embedded Teamcenter EDA library operations

The embedded EDA library connector inserts a Teamcenter menu in the menu bar of the EDA tool and adds a menu item for each EDA operation. If integrating into the tools menu is not possible, you can add a pop-up dialog, for example, using menu push-buttons. Each of these menu items triggers the custom connector code to support the following EDA operations:

- Export Library
- Synchronize Library
- Save Library
- Check out Library
- Check in Library
- Cancel a checked out Library
- Revise Library
- Load Library
- Set a working Library

- Configure Library
- Initiate part workflow
- Check the part workflow status
- Obtain user workflow tasks using getMyWorklist

Export Library

You can export the entire EDA library data into Teamcenter. The connector code performs the following steps to implement this operation.

1. Create an **EDALib.xml** file (see exportLibrary API example) with all part information, attributes, and the symbol or footprint for the shape library that is to be exported into Teamcenter.

2. Create an empty file to receive operation status from the EDA common client. This is the EDA Status file.

3. Call the **exportLibrary** API.

   The EDA common client processes the information in the **EDALib.xml** file as follows:

   a. Create or update a part, symbol, or footprint into Teamcenter.

   b. Create or update classification objects for parts if parts have category information.

   c. Map the attributes to the parts or its classification object based on attribute mapping configuration.

   d. Create or update the dataset in Teamcenter with the provided dataset types, relation, and primary object information in the **EDALib.xml** file.

   e. Link all the library data with the Teamcenter external application site. Ensure that the part properties from the Teamcenter rich client and the **Exported to** property have the value of the Tc Part Library (external application name) defined in working configuration as follows:

4. Check EDAExportLibraryStatus in the status file returned from the call.

   a. Abort if **status** != “Success”.

Synchronize Library

You can synchronize the latest changes in Teamcenter with the ECAD tool since the last synchronization operation. The connector code performs the following steps to implement this operation:

1. Create an empty **EDALib.xml** file to hold synchronization information loaded from Teamcenter.

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

3. Call the **syncLibrary** API. EDA common client does the following synchronize objects:

   a. Find new parts,using the saved query defined in the **EDALIB_Sync_FindNewParts** preference. Each search criterion in a saved query must contain a default value, and the EDA common client does not support setting the value at runtime.

   b. Find the latest versions of the modified parts in the Teamcenter external application site. The EDA common client uses the TIE synchronizer framework to find the latest modified parts. TIE synchronizer uses the ImanExportRecord object which compares the last export date attribute with the last modified date of an object to decide if the object has been modified or not since the last export date. For more details about the TIE synchronizer, refer to the Teamcenter Server Customization help on Support Center.

      Only objects with the **Export to** attribute point to the working configuration Teamcenter part library referenced by an ImanExportRecord and it are considered by TIE synchronizer.

   c. Filter the candidates for synchronization against the new or modified parts based on the saved query defined in the **EDALIB_Sync_FilterSyncParts** preference. The objects that are returned as search results after steps a and b form the input of the filter saved query.

   d. Validate all the candidates. The EDA common client finally checks the output in step c. If any object and its associated object (BOM revision, primary dataset, or item revision) are checked out, the EDA common client ignores the object during synchronization.

   e. Synchronize all the valid parts. It uses the manage rule to load synchronization objects from Teamcenter into an **EDALib.xml** file in step 1 and then uses TIE SOA to update the ImanExportRecord last export date.

4. Check the **EDAImportLibraryStatus** in the status file returned from the call.

   Abort if **status** != “Success”.

5. The new library connector reads the **EDALib.xml** file and then imports the data into the ECAD library.

Save Library

Users can save new or checked out parts into Teamcenter from the ECAD side.

1. Create an **EDALib.xml** file with part information, attributes as well as the symbol and footprint for a shape library that is to be saved to Teamcenter.

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

3. Call the **saveLibrary** API. The parts saved into Teamcenter via the saveLibrary API do not link with any Tc Part Library (external application). They are generally called as new parts.

4. The first synchronization of the new parts depends on saved query defined in **EDALIB_Sync_FindNewParts** to find parts of each kind. Once the part is synchronized, it will reference with ImanExportRecord and the TIE synchronizer subsequently manages the modification.

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

   Abort if **status** != “Success”.

Check out Library

Users can load the existing data from Teamcenter into the ECAD side and then check out the given objects in the EDALib.xml file.

1. Create an empty **EDALib** XML file to hold the synchronization information from Teamcenter.

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

3. Call the **checkOutLibraryAPI**. The EDA Common Client checks out the library as follows:

   a. Load the existing data from Teamcenter with the given item IDs in the **EDALib.xml** file.

   b. Check out objects and its associated objects (item revision, BOM revision, item revision master form, and primary dataset, if applicable).

4. Check the **EDALibStatus** in the status file returned from the call.

   Abort if **status** != “Success”.

5. The new library connector reads the **EDALib.xml** file and then imports the data into the ECAD library.

Check in Library

Users can save information in the **EDALib.xml** file and then check in the given objects in Teamcenter. The connector code performs the following steps to implement this operation:

1. Create an **EDALib.xml** file with part information, attributes as well as the symbol and footprint for a shape library to be saved and checked in to Teamcenter.

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

3. Call the **checkInLibraryAPI**. EDA common client process checked-in operation as follows:

    a. Call the saveLibrary operation to save all the information in the **EDALib.xml** file into Teamcenter.

    b. Check in objects and its associated objects (item revision, BOM revision, item revision master form, and primary dataset, if applicable)

4. Check the EDALibStatus in the status file returned from the call.

    Abort if **status** != “Success”.

Cancel a checked out Library

You can cancel the checkout operation in Teamcenter. The connector code performs the following steps to implement this operation:

1. Create an empty **EDALib.xml** file to hold the synchronization information load from Teamcenter.

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

3. Call the **cancelCheckOutLibrary** API. EDA common client process checked-in operation like below.

    a. Cancel the checked out objects in Teamcenter.

    b. Reload the objects from Teamcenter into the ECAD side to a clean local modification.

4. Check the **EDALibStatus** in the status file returned from the call.

    Abort if **status** != “Success”.

5. A new library connector reads the **EDALib.xml** and then imports the data into the ECAD library.

Revise Library

You can revise the objects in Teamcenter. The connector code performs the following steps to implement this operation:

1. Create an empty **EDALib.xml** file to hold synchronization information from Teamcenter.

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

3. Call the **reviseLibrary** API. EDA common client processes the check in operation as follows.

a. Open a dialog to obtain user inputs.

b. Revise the objects.

c. Check out the latest revision, in case of user selections, and load the latest revision into **EDALib.xml** file in step1.

4. Check the EDALibStatus in the status file returned from the call.

   Abort if **status** != “Success”.

A new library connector reads the **EDALib.xml** and then imports data into the ECAD library if **check-out** is selected during revising.

Load Library

Users can load valid parts from Teamcenter into the ECAD side. For instance, if a designer wants to load a part in library 1 to library 2, from Teamcenter. The connector code performs the following steps to implement this operation:

1. Create an empty **EDALib.xml** file to hold the synchronization information from Teamcenter.

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

3. Call the **loadLibrary** API. EDA common client processes load library operation as follows:

   a. In the load library dialog, select the valid parts to load.

   b. Validate the user selected parts. These must be an item or item revision for a shape library and the selected object must contain a primary dataset under the item revision.

   c. Load valid parts from Teamcenter into the **EDALib.xml** file created in step1.

4. Check the EDALibStatus in the status file returned from the call.

   Abort if **status** != “Success”.

A new library connector reads the **EDALib.xml** and then imports the data into the ECAD library

Set a working library

Users can set a working library outside of the EDA library gateway. All EDA library operations such as export library and display are based on the working library settings. The EDA common client reads the current working library and displays all the parts in this working library. The connector code performs the following steps to implement this operation:

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

2. Call the **setWorkingLibrary** API with a specific configuration name (see setWorkingLibrary API).

3. Check the EDALibStatus in the status file returned from the call.

   Abort if **status** != “Success”.

Configure Library

Users can create a library configuration outside of the EDA library gateway directly from the embedded connector. The connector performs the following steps to implement this operation:

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

2. Call the **configureLibrary** API to return a dialog for the configuration (see configureLibrary API).

3. Check the EDALibStatus in the status file returned from the call.

   Abort if **status** != “Success”.

   Note: This eliminates the need to open an EDA library gateway first to create the configuration by the **Teamcenter** > **Configuration** > **Synchronization** tab route.

Initiate part workflow

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

1. Create an **EDADesign.xml** file with all the BOM information in the file. The BOM can be a base BOM or variant BOM.

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

3. Call the **initiateWorkflow** API to initiate the workflow. EDA common client process check in operation as follows:

   a. A workflow wizard is executed.

   b. If EDADesign xml file is passed into parameter, the EDA common client extracts the BOM information and list all parts into both **Target** and **Reference** pages.

   c. You can add or modify the workflow information on the wizard.

      Click the **Finish** button.

The EDA common client creates a workflow in Teamcenter.

4. Check the EDALibStatus in the status file returned from the call.

   Abort if **status** != “Success”.

Check the part workflow status

You can view the part workflow status on the completed tasks, pending tasks, and ongoing tasks of the workflow in a graphic viewer on the ECAD side. The connector code performs the following steps to implement this operation:

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

2. Call the **checkPartWorkflowStatus** API.

   You can obtain an overview of the workflow status and view detailed information like responsible party and task status for a selected task on the viewer

3. Check the **EDALibStatus** in the status file returned from the call.

   Abort if **status** != “Success”.

Obtain user workflow tasks using getMyWorkslist

Users can view their current workflow tasks. The connector code performs the following steps to implement this operation:

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

2. Call the **getMyWorklist** API to obtain all the DoTask and Perform Signoff workflow tasks for a user.

   The user can now view their workflow status and perform actions for the workflow task.

3. Check the **EDALibStatus** in the status file returned from the call.

   Abort if **status** != “Success”.

Customizing application types

Customize application type

The EDA common client supports custom definitions of your own application type in the “Application Type” field in the External Application section of Teamcenter Organization application.

Add a customized application type

Create the **EDALIB_External_Applications** preference and add the customized application type as shown below:

```
EDALIB_External_Applications=<application>,<application integer
type>[,<Option>]
```

- **application** - This is the customized application name, and it should be the same as application name defined in the <application>_edadef.xml configuration file. This name will be displayed in the **Application Type** box in the **External Application** of Teamcenter Organization application.

- **application integer type** - it is an integer value representing the application type in the Teamcenter database. Each application integer type should be unique.

- **option** - this value is mandatory only for Shape Library. It defines an option name that is used to synchronize a primary dataset during synchronization option. Examples of existing option names include **opt_expedition_ds** in **TIEEDALibExport** Transfer OptionSet of PLM XML or TC XML Export Import Administration.

Enable option

This is only applicable to Shape Library integrations. Using the existing **opt_expedition_ds** option in PLM XML or TC XML Export Import Administration as an example, enable the option defined in the preference.

- Add the option in **TIEOptionSetEdaLib** of TransferOptionSet.

- Enable the option in **TIEEDALibExport** of ClosureRule.

Create a customized external application

Users can create customized external applications from the rich client. The application name is available in the **Application Type** combo box, using which users can create the site in Teamcenter.

The current application type display name does not support localization.

5. EDALib

EDALib

The information between an EDA connector and the EDA common client is passed in one of two ways. The connector creates this file and passes it to the common client through the API call or it creates an empty file in which the common client writes the information and passes it back to the connector. In such cases, the connector parses the file to obtain information.

The schema file is available in the *%TCEDA_ROOT%***\example\schema\EDALib.xsd**.

A sample **EDALib** file outline is shown here. Individual types are described in the subsequent topics in this section.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id17" exportDate="2013-06-09+08:00"
            folder="D:\LibraryTest\library"/>
    <item elemId="id18" name="comp01_1" libraryId="id17"
itemId="comp01_1"
            type="Component" category="Electronic
Parts,RESISTOR,FIXED,LINEAR"
            childList="id19 id22 id25">
        <attr name="Cost" value="10" type="String"/>
        <attr name="Tolerance" value="34%" type="String"/>
    </item>
    <item elemId="id19" name="comp01_1sym01_1" libraryId="id17"
            itemId="comp01_1sym01_1" type="Symbol"
            category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="id22" name="comp01_1sym01_2" libraryId="id17"
            itemId="comp01_1sym01_2" type="Symbol"
            category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="id25" name="comp01_1ft01_1" libraryId="id17"
            itemId="comp01_1ft01_1" type="Footprint"
            category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <gde elemId="id28" name="comp01_1ft01_1ps01_1" libraryId="id17"
            type="Padstack"/>
    <relation elemId="id31" type="EDAPadstackRelation" primary="id25"
            secondary="id28"/>
    <relation elemId="id33" type="IMAN_reference" primary="id18"
secondary="id32"/>
    <relation elemId="id197" type="IMAN_reference" primary="id18"
            secondary="id196"/>
    <dataset elemId="id32" name="comp01_1cadencelib1" type="cadenceLib"
            folder="D:\Workdir\test\edalib\datasets\comp01_1"/>
    <dataset elemId="id196" name="comp01_10text2" type="Text"
```

```
                folder="D:\Workdir\test\edalib\datasets\comp01_10"/>
</edaLib>
```

libraryType

The **EDALib** must contain one **libraryType** element.

```
<library elemId="id1" folder="D:\LibraryTest\library"/>
```

Attribute name	Use	Description
elemId	Required	Unique object id.
name	Optional	Name of the ECAD part library.
exportDate	Optional	Date of the export. This is used only when the EDA common client updates the **EDALib.xml** file.
folder	Optional	Library path for shape library integrations.
targetSiteId	Optional	Teamcenter external application site id.

itemType

All objects that are described in this element are represented as an item in Teamcenter.

```
<item elemId="id2" name="comp01_1" libraryId="id1" itemId="comp01_1"
 type="Component" category="Electronic Parts,RESISTOR,FIXED,LINEAR"
 childList="id3 id6">
 <attr name="Cost" value="10" type="String"/>
 <attr name="Tolerance" value="34%" type="String"/>
</item>
```

Attribute name	Use	Description
elemId	Required	Unique element id
name	Required	Item name of the EDA part created or updated in Teamcenter
libraryId	Required	
itemId	Required	ItemId of the EDA part created or updated in Teamcenter
Revision	Required	
type	Required	The object type name in the ECAD tool, such as component, symbol, or footprint. See the preference EDALIB_ItemTypesMapping definition in Teamcenter.
category	Optional	Category name of the EDA part
childList	Optional	The element ID points to other objects that will be added into the BOMRevision view for the current item revision.

Attribute name	Use	Description
	Deprecated	
label	Optional	
revLabel	Optional	
releaseStatus	Optional	
cadAttrNames	Optional	
cadAttrValues	Optional	

gdeType

All objects that are described in this element are represented as an item in Teamcenter. **EDALib** may contain one or more **shapeType** elements.

```
<gde elemId="id9" name="comp01_1ft01_1ps01_1" libraryId="id1"
type="Padstack"/>
```

Element name	Use	Description
elemId	Required	Unique element id
name	Required	The GDE object name created or updated in Teamcenter.
libraryId	Required	
type	Required	The object type name in the ECAD tool, for example, padstack. See the preference EDALIB_GDETypesMapping definition in Teamcenter.
label	Optional	
releaseStatus	Optional	

datasetType

The **EDALib** may contain one or more **datasetType** elements. For shape library integrations, the dataset must be specified for the part, symbol, footprint, or padstack object.

```
<dataset elemId="id4" name="comp01_1sym01_1cadencelib1"
type="mentorExpLib"
 folder="D:\Workdir\test\edalib\datasets\comp01_1sym01_1"/>
<dataset elemId="id15" name="comp01_1text2" type="Text"
 folder="D:\Workdir\test\edalib\datasets\comp01_1"/>
```

Element name	Use	Description
elemId	Required	Unique element id
name	Required	Dataset name
type	Required	Dataset type name in the ECAD side. See the preference EDALIB_DatasetTypesMapping defined in Teamcenter. If the EDA common client cannot find a mapped dataset type from the preference, it uses the specified value to create a dataset.
label	Optional	
folder	Optional	The folder path that represents the named reference for the dataset.
cadAttrNames	Optional	
cadAttrValues	Optional	

relationType

This element defines the relation between two objects in Teamcenter.

```
 <relation elemId="id11" type="IMAN_reference" primary="id9"
secondary="id10"/>
 <relation elemId="id12" type="EDAPadstackRelation" primary="id6"
secondary="id9"/>
```

Element name	Use	Description
elemId	Required	Unique element id
primary	Required	Primary element ID of the relation.
secondary	Required	Secondary element ID of the relation.
label	Optional	
type	Optional	ImanRelation type. It must be a real name defined in BMIDE.

attrType

The **EDALib** may contain one or more **attrType** element for part attributes.

```
<attr elemId="id28" name="Cost" value="10" type="String"/>
```

Element name	Use	Description
elemId	Required	Unique element id
name	Required	Attribute name
value	Required	Attribute value

Element name	Use	Description
type	Optional	Attribute value type defined in attrValueType
unit	Optional	Attribute unit. This is used for unit conversion. For example, if the length of the unit in ECAD attribute is **meter** and that in Teamcenter classification **centimeter**, the value will be converted to **centimeter** when saving the value in Teamcenter.
required	Optional	Specifies whether an attribute is mandatory during item creation. The value must be set to **true**.

attrValueType

A valid attribute value type is as follows:

```
<xs:restriction base="xs:string">
<xs:enumeration value="Integer"></xs:enumeration>
<xs:enumeration value="Double"></xs:enumeration>
<xs:enumeration value="Date"></xs:enumeration>
<xs:enumeration value="Boolean"></xs:enumeration>
<xs:enumeration value="String"></xs:enumeration>
<xs:enumeration value="Reference"></xs:enumeration>
<xs:enumeration value="ReferencedValue"></xs:enumeration>
```

Element name	Description
Reference	The elemId of a different element in the EDALib xml file. The EDA common client creates a reference object and then use the UID to create or update the item object.
ReferenceValue	The UID of an existing object in Teamcenter. The EDA common client uses specified UID and sets the property of the item object during creation or modification.

6. EDALibConf

EDALibConf

The information describing an ECAD tool library is stored internally in this schema. This describes the configuration of the library.

EDALibConf is the main element of this schema, there must be one of this at the top level.

Attribute name	Use	Description
name	Required	The name of the library configuration. This is specified in the Configuration dialog as “Library Configuration”
enableECADLibrary	Required	This boolean attribute specifies whether synchronization to the ECAD library is enabled. This is controlled by the Configuration dialogs “Configure ECAD Library” button.
partMetadataDBName	Optional	This is the path of the metadata database configuration. This is from the “Configure” sub-dialog of the Configuration dialog.
enableMetadataDB	Required	This boolean attribute specifies whether synchronization to the part metadata database is enabled. This is controlled by the Configuration dialogs “Configure Part Metadata Database” button.
enableFileSync	Required	This boolean attribute specifies whether synchronization via manual file transfer is enabled. This is controlled by the Configuration dialogs “Configure File Mode” button.
syncFolder	Optional	This string attribute specifies the folder to create the sync file (from Teamcenter) in by default.
description	Optional	The description of the library configuration. This is not currently used.

The **EdaLibConf** may contain the different sub-elements as described in this section.

EcadLibrary

This element represents the ECAD library.

Attribute name	Use	Description
libraryPath	Required	The path to the installed ECAD library.
setReadOnly	Optional	This boolean specifies whether the ECAD library is read-only, true by default. This value is specified by the “Set Data Source Folder as Read Only” button on the Configuration dialog.

Attribute name	Use	Description
exportGraphics	Optional	This boolean specifies whether parts graphical entities shall be exported, true by default. This value is specified by the “Symbol and Footprint” button on the Configuration dialog (currently not used).
exportAttributes	Optional	This boolean specifies whether parts attributes shall be exported, true by default. This value is specified by the “Part Attributes” button on the Configuration dialog (currently not used).

TcPartLibrary

This element represents the Teamcenter part library.

Attribute name	Use	Description
libraryId	Required	The library Id from the “TcPart Library” combo-box on the Configuration dialog.
libraryName	Required	The library name from the “TcPart Library” combo-box on the Configuration dialog.

EcadObjectMapping

Attribute name	Use	Description
ecadType	Required	EDA object type
tcType	Required	Teamcenter object type
item	Optional	This boolean specifies whether tcType is type or subtype of Item type in Teamcenter, true by default.

ManagedObject

Attribute name	Use	Description
primaryType	Required	Primary object type
Name	Required	Relation name between primary object type and secondary object type
displayValue	Required	Display value for Relation name
secondaryTypes	Required	Secondary object type

7. Status file

The status file is used to return results and the status from the EDA common client API to the EDA connector. The schema is defined in chapter **EDAStatus** of the **Schema Reference** document. The schema file is available in the *%TCEDA_ROOT%***\example\schema\EDAStatus.xsd**.

8. Common Client API

Overview of common client APIs

The common client methods may be accessed through a command-line interface.

The command-line interface is provided through a Windows batch file (**edacli.bat**) or a Unix script file (**edacli**). This script launches a java process, passing the relevant arguments. Before calling this command line, make sure that you run the **setup_eda** and **setup_ecadlibrary** scripts to set up a minimalistic environment. Since each call starts and exits a new process, you must ensure the following:

- The application type must be passed in on each call.
- The Teamcenter login session must be reestablished for each call. This is done inside the common client, using cookies.

All arguments are input only.

Many of the interface methods use the same arguments, and they are as follows:

- **applicationName** – The application name representing the ECAD tool in use. This value specifies which configuration file to use.
- **edaLib** – An absolute pathname to an EDALib file containing information that is either passed from the connector to the common client or used by the common client to formulate the contents of the file. Make sure that the edaLib file is created and ready before you send it to the EDA common client.
- **status** – An absolute path for an EDAStatus file that is created by the common client to return the status to the connector. Make sure that the status file is created and ready before you send it to EDA common client.

exportLibrary

This API is called to export all data in the EDALib file into Teamcenter. The common client creates or updates the related objects in Teamcenter.

Command line API

```
edacli
    –exportLibrary
    –application applicationName
    –mode        mode
    –pf          passwordFile
    -scope       scope
```

```
–edaLib      EDALib file
–status      status file
```

- Mode – Export library has two modes, default value is 0:
  - 0 - Automated mode. Mainly used when synchronize library from Teamcenter at a schedule time like Windows task scheduler, for this mode common-client uses user/password information defined in TcEDAClient.properties to login Teamcenter automatically without any user interaction.
  - 1 - On Demand mode. Invoked by user action, like click menu, button on gateway UI or as API input, this needs user action to manually login if Teamcenter connection has not been establish.
- Pf – Password file. This must be a reference to an encrypted file containing the password for the default user account (specified in TCEDAClient.properties).
- Scope – The scope option determines which items or objects are included in the export.

  It can take one of the following values:
  - **All**: This value includes all available items or objects, regardless of whether they are new or already existing in Teamcenter.
  - **Exisiting**: This value includes items or objects that already exist in Teamcenter.
  - **New**: This value includes items or objects that do not exist in Teamcenter.

  If it is not specified, the scope is considered as **ALL**.

Command line example

```
edacli.bat -exportLibrary -application mentorEDXLib -mode 0 -edaLib
D:\EDALib\edx\edx_edalib.json -status D:\EDALib\edx\edx_status.xml -pf
D:\pass_ed.txt -scope ALL
```

EDALib file example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id17" exportDate="2013-06-09+08:00"
            folder="D:\LibraryTest\library"/>
    <item elemId="id18" name="comp01_1" libraryId="id17"
itemId="comp01_1"
            type="Component" category="Electronic
Parts,RESISTOR,FIXED,LINEAR"
            childList="id19 id22 id25">
        <attr name="Cost" value="10" type="String"/>
```

```
        <attr name="Tolerance" value="34%" type="String"/>
    </item>
    <item elemId="id19" name="comp01_1sym01_1" libraryId="id17"
            itemId="comp01_1sym01_1" type="Symbol" category="Electronic
            Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="id22" name="comp01_1sym01_2" libraryId="id17"
            itemId="comp01_1sym01_2" type="Symbol" category="Electronic
            Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="id25" name="comp01_1ft01_1" libraryId="id17"
            itemId="comp01_1ft01_1" type="Footprint" category="Electronic
            Parts,RESISTOR,FIXED,LINEAR"/>
    <gde elemId="id28" name="comp01_1ft01_1ps01_1" libraryId="id17"
            type="Padstack"/>
    <relation elemId="id31" type="EDAPadstackRelation" primary="id25"
            secondary="id28"/>
    <relation elemId="id33" type="IMAN_reference" primary="id18"
secondary="id32"/>
    <relation elemId="id197" type="IMAN_reference" primary="id18"
            secondary="id196"/>
    <dataset elemId="id32" name="comp01_1cadencelib1" type="cadenceLib"
            folder="D:\Workdir\test\edalib\datasets\comp01_1"/>
    <dataset elemId="id196" name="comp01_10text2" type="Text"
            folder="D:\Workdir\test\edalib\datasets\comp01_10"/>
</edaLib>


{
    "AllContainers": {
        "toolIdentifier": "MENTOREXPLIB",
        "skipPropertyInflation": "True",
        "Containers": [
            {
                "containerIdentifier": "id2",
                "tcContainerType": "",
                "tcLastModifiedDate": "",
                "containerName": "12301-CAP",
                "containerType": "Component",
                "tcContainerName": "",
                "action": "add",
                "tcFolderUid": "",
                "isTopContainer": true,
                "tcReservation": false,
                "attributes": [
                    {
                        "tcAttrName": "",
                        "attrName": "id",
                        "value": [ "12301-CAP" ]
                    },
                    {
```

```
                        "tcAttrName": "",
                        "attrName": "name",
                        "value": [ "12301-CAP" ]
                    },
                    {
                        "tcAttrName": "",
                        "attrName": "classificationCategory",
                        "value": [ "CAPACITOR" ]
                    }
                ]
            }
        ]
    }
}
```

Status file example

This has been enhanced to show warnings and errors reported from Teamcenter server.

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

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDAExportLibraryStatus status="Error" message="Cannot construct an
ModelObject for Q5VNDXVP61UEEC. The type Briefcase does not exist in the
Client Meta Model."
        xmlns="http://www.ugs.com/tc/EDAStatus"/>
```

syncLibrary

This API is called to synchronize EDA parts from Teamcenter into the ECAD side.

Command line API

```
edacli
    –syncLibrary
    –application applicationName
    -mode        Synchronozation mode
    –pf          passwordFile
    -workingLibraryName configuration name.
    –edaLib      empty EDALib file
    –status      status file
```

- Mode - Synchronize library have two modes, default value is 1:
  - 0 - Automated mode. Mainly used when synchronize library from Teamcenter at a schedule time like Windows task scheduler, for this mode common-client uses user/password information defined in TcEDAClient.properties to login Teamcenter automatically without any user interaction.
  - 1 - On Demand mode. Invoked by user action, like click menu, button on gateway UI or as API input, this needs user action to manually login if Teamcenter connection has not been establish.
- Pf – Password file. This must be a reference to an encrypted file containing the password for the default user account (specified in TCEDAClient.properties).
- workingLibraryName – Configuration name that is created via gateway Teamcenter -> Configure -> Synchronization tab. If no value is provided common-client uses current working configuration name to synchronize library.

Command line example

```
edacli.bat –syncLibrary –application  mentorExpLib –edaLib
C:\temp\edalib.xml _
    –status C:\temp\status.xml
```

or

```
edacli.bat –syncLibrary –application  mentorExpLib -mode 1 _
    -workingLibraryName exp_config1  -edaLib C:\temp\edalib.xml _
    –status C:\temp\status.xml
```

Status file example

This has been enhanced to show warnings and errors from the Teamcenter server.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EDAExportLibraryStatus status="Success" xmlns="http://www.ugs.com/tc/
EDAStatus">
    <Warnings>
        <warning>    Code: 249502    Severity: 2    3D model
"User:YAGEO_RC0402FR-070RL" found with non-zero alignment. Component
"Comp 1" using this model was not exported.</warning>
        <warning>    Code: 249503    Severity: 2    Component "AS002"
has denied "WRITE" access. This component was not exported.</warning>
    </Warnings>
</EDAExportLibraryStatus>
```

checkoutLibrary

This operation allows the user to load the existing data from Teamcenter into the ECAD side and then check out the given objects in the EDALib xml file.

Command line API

```
edacli
    –checkoutLibrary
    –application     applicationName
    –selectedObjects EDALib file which contains selected objects to be
checked out
    –edaLib          empty EDALib file which holds information loaded
from
                     Teamcenter for checked-out objects
    –status          status file
```

Command line example

```
edacli.bat –checkoutLibrary –application  mentorExpLib _
    -selectedObjects C:\temp\ selectededalib.xml –edaLib
C:\temp\edalib.xml _
    –status C:\temp\status.xml
```

EDALib file example

- **SelectedObjects file**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
            libraryId="id111" itemId="comp01_1" type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5b" name="comp01_10"
            libraryId="id111" itemId="comp01_10" type="Component"/>
</edaLib>
```

- **Status file example**

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

saveLibrary

This attribute is obsolete in ECAD library integration.

checkInLibrary

This operation allows the user to save information in the **EDALib.xml** file and then check in the given objects in Teamcenter.

Command line API

```
edacli
–checkInLibrary
–application applicationName
–selectedObjects   EDALib file which contains selected objects to be
check in
–edaLib   EDALib file which holds related information to update checked
in objects
–status      status file
```

Command line example

```
edacli.bat –checkInLibrary –application  mentorExpLib -selectedObjects
C:\temp\ selectededalib.xml –edaLib C:\temp\edalib.xml –status
C:\temp\status.xml
```

EDALib file example

- **SelectedObjects file**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
libraryId="id111" itemId="comp01_1" type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5b" name="comp01_10"
libraryId="id111" itemId="comp01_10" type="Component"/>
</edaLib>
```

- **EDALib file**

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

```
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id114" folder="D:\LibraryTest\library"/>
    <item elemId="44956e0bde6a4b70b12b62250b89c79c"
name="comp01_10sym01_2"
    libraryId="id114" itemId="comp01_10sym01_2" type="Symbol"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="096a02e0aa97477385985d0edda49266"
name="comp01_1sym01_1"
    libraryId="id114" itemId="comp01_1sym01_1" type="Symbol"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="c36e07a9750247e59589ac6dc8c6a1e9"
name="comp01_10ft01_1"
    libraryId="id114" itemId="comp01_10ft01_1" type="Footprint"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="c304c1be482c489c80a7a93163b77a3c"
name="comp01_1ft01_1"
    libraryId="id114" itemId="comp01_1ft01_1" type="Footprint"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="03601b67341343b4b524db8aa7a2825a" name="comp01_1"
    libraryId="id114" itemId="comp01_1" type="Component"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"
    childList="096a02e0aa97477385985d0edda49266
58c8f90e2c5646a3ab1d85d0948ce245
    c304c1be482c489c80a7a93163b77a3c">
        <attr name="Cost" value="10" type="String" required="false"/>
        <attr name="Tolerance" value="34%" type="String"
required="false"/>
    </item>
    <item elemId="58c8f90e2c5646a3ab1d85d0948ce245"
name="comp01_1sym01_2"
    libraryId="id114" itemId="comp01_1sym01_2" type="Symbol"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <item elemId="92808e6e566a44c2845b4d1dac378bd6" name="comp01_10"
    libraryId="id114" itemId="comp01_10" type="Component"
    category="Electronic Parts,RESISTOR,FIXED,LINEAR"
    childList="4d3ab922022f4f2e9dbb8c3a97307224
44956e0bde6a4b70b12b62250b89c79c
    c36e07a9750247e59589ac6dc8c6a1e9">
        <attr name="Cost" value="10" type="String" required="false"/>
        <attr name="Tolerance" value="34%" type="String"
required="false"/>
    </item>
    <item elemId="4d3ab922022f4f2e9dbb8c3a97307224"
name="comp01_10sym01_1"
    libraryId="id114" itemId="comp01_10sym01_1" type="Symbol" category=
"Electronic Parts,RESISTOR,FIXED,LINEAR"/>
    <gde elemId="b26d951199144b0dba58a1fe57c699f9"
name="comp01_1ft01_1ps01_1"
    libraryId="id114" type="Padstack"/>
```

```
    <gde elemId="c79d2059aa3b4ceba5de53c4c178e45e"
name="comp01_10ft01_1ps01_1"
    libraryId="id114" type="Padstack"/>
    <relation elemId="id117" type="IMAN_reference"
    primary="b26d951199144b0dba58a1fe57c699f9"
    secondary="369946e38b8b4a3ba9c1abba6f1e5ab9"/>
    <relation elemId="id120" type="IMAN_reference"
    primary="44956e0bde6a4b70b12b62250b89c79c"
    secondary="38168fecf2284580b1369435353209ac"/>
    <relation elemId="id121" type="IMAN_reference"
    primary="096a02e0aa97477385985d0edda49266"
    secondary="05e5b4f603ab466aa51f661020c8f6aa"/>
    <relation elemId="id123" type="IMAN_reference"
    primary="c79d2059aa3b4ceba5de53c4c178e45e"
    secondary="89697e16f9be406392cab56601942801"/>
    <relation elemId="id125" type="IMAN_reference"
    primary="c36e07a9750247e59589ac6dc8c6a1e9"
    secondary="657c9633468d43868da792b4ded6f09e"/>
    <relation elemId="id129" type="EDAPadstackRelation"
    primary="c36e07a9750247e59589ac6dc8c6a1e9"
    secondary="c79d2059aa3b4ceba5de53c4c178e45e"/>
    <relation elemId="id132" type="IMAN_reference"
    primary="c304c1be482c489c80a7a93163b77a3c"
    secondary="0280a6cf000e4cb696fc8783bb9970ac"/>
    <relation elemId="id135" type="EDAPadstackRelation"
    primary="c304c1be482c489c80a7a93163b77a3c"
    secondary="b26d951199144b0dba58a1fe57c699f9"/>
    <relation elemId="id138" type="IMAN_reference"
    primary="03601b67341343b4b524db8aa7a2825a"
    secondary="5f89733da6504489bfa9c1b3964c270b"/>
    <relation elemId="id139" type="IMAN_reference"
    primary="03601b67341343b4b524db8aa7a2825a"
    secondary="4f18e62110014096a8f461c3129af4f7"/>
    <relation elemId="id141" type="IMAN_reference"
    primary="58c8f90e2c5646a3ab1d85d0948ce245"
    secondary="0d1d401f52e5456a8e374b06f4d73994"/>
    <relation elemId="id143" type="IMAN_reference"
    primary="92808e6e566a44c2845b4d1dac378bd6"
    secondary="25fbd37260684789815b203c582d4242"/>
    <relation elemId="id147" type="IMAN_reference"
    primary="92808e6e566a44c2845b4d1dac378bd6"
    secondary="d434736adc1f46929f084326c9a2a925"/>
    <relation elemId="id150" type="IMAN_reference"
    primary="4d3ab922022f4f2e9dbb8c3a97307224"
    secondary="f6e13f8ecd564300a3c6f913740960a4"/>
    <dataset elemId="f6e13f8ecd564300a3c6f913740960a4" name=
    "comp01_10sym01_1cadencelib1" type="cadenceLib" folder=
    "D:\Workdir\test\edalib\datasets\comp01_10sym01_1"/>
    <dataset elemId="369946e38b8b4a3ba9c1abba6f1e5ab9" name=
```

```
        "comp01_1ft01_1ps01_1cadencelib1" type="cadenceLib" folder=
        "D:\Workdir\test\edalib\datasets\comp01_1ft01_1ps01_1"/>
        <dataset elemId="0d1d401f52e5456a8e374b06f4d73994" name=
        "comp01_1sym01_2cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_1sym01_2"/>
        <dataset elemId="5f89733da6504489bfa9c1b3964c270b" name=
        "comp01_1cadencelib1" type="cadenceLib" folder=
        "D:\Workdir\test\edalib\datasets\comp01_1"/>
        <dataset elemId="05e5b4f603ab466aa51f661020c8f6aa" name=
        "comp01_1sym01_1cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_1sym01_1"/>
        <dataset elemId="89697e16f9be406392cab56601942801" name=
        "comp01_10ft01_1ps01_1cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_10ft01_1ps01_1"/>
        <dataset elemId="657c9633468d43868da792b4ded6f09e" name=
        "comp01_10ft01_1cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_10ft01_1"/>
        <dataset elemId="25fbd37260684789815b203c582d4242" name=
        "comp01_10cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_10"/>
        <dataset elemId="38168fecf2284580b1369435353209ac" name=
        "comp01_10sym01_2cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_10sym01_2"/>
        <dataset elemId="0280a6cf000e4cb696fc8783bb9970ac" name=
        "comp01_1ft01_1cadencelib1" type="cadenceLib" folder="
        D:\Workdir\test\edalib\datasets\comp01_1ft01_1"/>
        <dataset elemId="d434736adc1f46929f084326c9a2a925" name=
        "comp01_10text2" type="Text" folder="
        D:\Workdir\test\edalib\datasets\comp01_10"/>
        <dataset elemId="4f18e62110014096a8f461c3129af4f7" name=
        "comp01_1text2" type="Text" folder="
        D:\Workdir\test\edalib\datasets\comp01_1"/>
    </edaLib>
```

- **Status file**

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

cancelCheckOutLibrary

This operation allows the user to cancel checked out objects in Teamcenter.

Command line API

```
edacli
–cancelCheckOutLibrary
–application applicationName
–selectedObjects   EDALib file which contains selected objects to be
cancel checked out
–edaLib   empty EDALib file to load information for cancel checked-out
objects.
–status      status file
```

Command line example

```
edacli.bat –cancelCheckOutLibrary –application  mentorExpLib
-selectedObjects C:\temp\selectededalib.xml –edaLib C:\temp\edalib.xml –
status C:\temp\status.xml
```

EDALib file example

- **SelectedObjects file**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
libraryId="id111" itemId="comp01_1" type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5b" name="comp01_10"
libraryId="id111" itemId="comp01_10" type="Component"/>
</edaLib>
```

- **Status file**

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

reviseLibrary

This operation allows the user to revise objects in Teamcenter.

Command line API

```
edacli
–reviseLibrary
–application applicationName
–selectedObjects   EDALib file which contains selected objects to be
cancel checked out
–edaLib   empty EDALib file to load revised object information if user
selects check-out option on the UI
–status      status file
```

Command line example

```
edacli.bat –reviseLibrary –application  mentorExpLib -selectedObjects
C:\temp\selectededalib.xml –edaLib C:\temp\edalib.xml –status
C:\temp\status.xml
```

EDALib file example

- **SelectedObjects file**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
libraryId="id111" itemId="comp01_1" type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5b" name="comp01_10"
libraryId="id111" itemId="comp01_10" type="Component"/>
</edaLib>
```

- **Status file example**

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

loadLibrary

This operation allows the user to load existing objects from Teamcenter into the ECAD side.

Command line API

```
edacli
–loadLibrary
–application applicationName
–selectedObjects Optional EDALib file which contains selected objects to
be loaded, no user interaction is required
-checkOutSelected Optional, checks out objects from the selectedObjects
file (true or false), works with the –selectedObjects file only
–edaLib   empty EDALib file which holds information loaded from
Teamcenter
–status      status file
```

Command line examples

```
edacli.bat –loadLibrary –application  mentorExpLib –edaLib
C:\temp\edalib.xml –status C:\temp\status.xml
edacli.bat –loadLibrary –application  mentorExpLib –selectedObjects
C:\temp\selectedObjs.xml –checkOutSelected true –edaLib
C:\temp\edalib.xml –status C:\temp\status.xml
```

EDALib input file example

- **SelectedObjects file**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
libraryId="id111" itemId="comp01_1" type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5c" name="comp02_10"
libraryId="id111" itemId="comp02_10" revision=”B” type="Component"/>
</edaLib>

For MultipleRevision enabled :
SelectedObjects file
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<edaLib xmlns="http://www.plmxml.org/Schemas/ecadlib">
    <library elemId="id111"/>
    <item elemId="5882dcc42db146f4b258d3406a9e2946" name="comp01_1"
libraryId="id111" itemId="comp01_1" revision=”C” type="Component"/>
    <item elemId="bce783d9b4b44e3aa85b0b16a2b6db5c" name="comp02_10"
libraryId="id111" itemId="comp02_10" revision=”B” type="Component"/>
</edaLib>
```

- **Status file**

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

setWorkingLibrary

This operation allows the user to set a working library outside of the EDA library gateway.

Command line API

```
edacli
–setWorkingLibrary
–application applicationName
–libraryConfigName Configuration name that is created via gateway
Teamcenter -> Configure -> Synchronization tab or created by
configureLibrary API.
–status      status file
```

Command line example

```
edacli.bat –setWorkingLibrary –application  mentorExpLib –
libraryConfigName exp1 –status C:\temp\status.xml
```

Status file example

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

configureLibrary

This operation allows the user to create a library configuration outside of the EDA library gateway.

Command line API

```
edacli
–configureLibrary
```

```
–application applicationName
–status      status file
```

Command line example

```
edacli.bat –configureLibrary –application  mentorExpLib
C:\temp\status.xml
```

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 allows the user to initiate a part workflow using an EDADesign XML file that contains design BOM information.

Command line API

```
edacli
–initiatePartWorkflow
   –application applicationName
   – edaDesign   EDA Design file which contains BOM information, all
parts in the BOM will be displayed on the
     workflow UI to allow to add into workflow targets/references.
–status      status file
```

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="sch" bom="false">
        <dataset type="schematic"
```

```
src="c:\ecad\staging\zhangl\zhangl_-2080945481\gateway\latest\gatewayTst0
03\sch"/>
        <CCAVariant variantName="dulux" bom="true">
            <component itemId="TS_R1K" name="Resister 1K" quantity="1">
                <rdn name="R2"/>
            </component>
            <component itemId="TS_IC7404H" name="7404HC" quantity="1">
                <rdn name="U3"/>
            </component>
            <component itemId="TS_CON2" name="Edge Connector"
quantity="1">
                <rdn name="J1"/>
            </component>
            <component itemId="TS_Q2N2222" name="Transister"
quantity="1">
                <rdn name="Q1"/>
            </component>
            <component itemId="TS_IC7432H" name="7432HC" quantity="1">
                <rdn name="U1"/>
            </component>
            <component itemId="TS_IC7408H" name="7408HC" quantity="2">
                <rdn name="U2"/>
                <rdn name="U4"/>
            </component>
            <component itemId="TS_R10K" name="Resister 10K" quantity="1">
                <rdn name="R1"/>
            </component>
        </CCAVariant>
        <CCAVariant variantName="economic" bom="true">
            <component itemId="TS_IC7408LS" name="7408LS" quantity="1">
                <rdn name="U4"/>
            </component>
            <component itemId="TS_IC7432LS" name="7432LS" quantity="1">
                <rdn name="U1"/>
            </component>
            <component itemId="TS_R1K" name="Resister 1K" quantity="1">
                <rdn name="R2"/>
            </component>
            <component itemId="TS_IC7404LS" name="7404LS" quantity="1">
                <rdn name="U3"/>
            </component>
            <component itemId="TS_Q2N2222" name="Transister"
quantity="1">
                <rdn name="Q1"/>
            </component>
        </CCAVariant>
        <CCAVariant variantName="standard" bom="true">
            <component itemId="TS_IC7408" name="Gate7408" quantity="2">
                <rdn name="U2"/>
```

```
                <rdn name="U4"/>
            </component>
            <component itemId="TS_R1K" name="Resister 1K" quantity="1">
                <rdn name="R2"/>
            </component>
            <component itemId="TS_IC7432" name="Gate7432" quantity="1">
                <rdn name="U1"/>
            </component>
            <component itemId="TS_IC7404" name="Gate7404" quantity="1">
                <rdn name="U3"/>
            </component>
            <component itemId="TS_Q2N2222" name="Transister"
quantity="1">
                <rdn name="Q1"/>
            </component>
            <component itemId="TS_R10K" name="Resister 10K" quantity="1">
                <rdn name="R1"/>
            </component>
        </CCAVariant>
    </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 shows the part workflow status on the user interface.

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
```

Status file example

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

getMyWorklist

This operation returns all workflow tasks for the current user.

Command line API

```
edacli
–getMyWorklist
   –application applicationName
–status      status file
```

Command line example

```
edacli.bat –getMyWorklist –application  mentorExpLib –status
C:\temp\status.xml
```

Status file example

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

9. Licensing

For any new Teamcenter EDA library connector integrations, the EDA common client only checks the basic license key, *ecad_gateway_all_lib*.