TEAMCENTER

Callbacks in Teamcenter EDA

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

EDA library integration callback reference

1. Using callbacks to customize the EDA client operations

During an EDA common client operation, you may want custom tasks to happen along with the main EDA operation. For example, you may want to upload a file created at run time using an ECAD tool command or download additional data related to the design during any Teamcenter operation. The EDA client does not perform such additional tasks apart from what is designed for any standard operation. You can use callbacks to add customized behavior to the EDA operations. The callbacks are the customization points provided by the EDA client through which additional customized operations can be added using the JAVA method or through scripts.

A callback will be called after it is defined and configured in the client or connector definition file (see **configuration** section). There are two types callbacks you may create and configure. Java callbacks are written in java and must be built into a package. Script callbacks are written in a command-line format and can be run within a console environment.

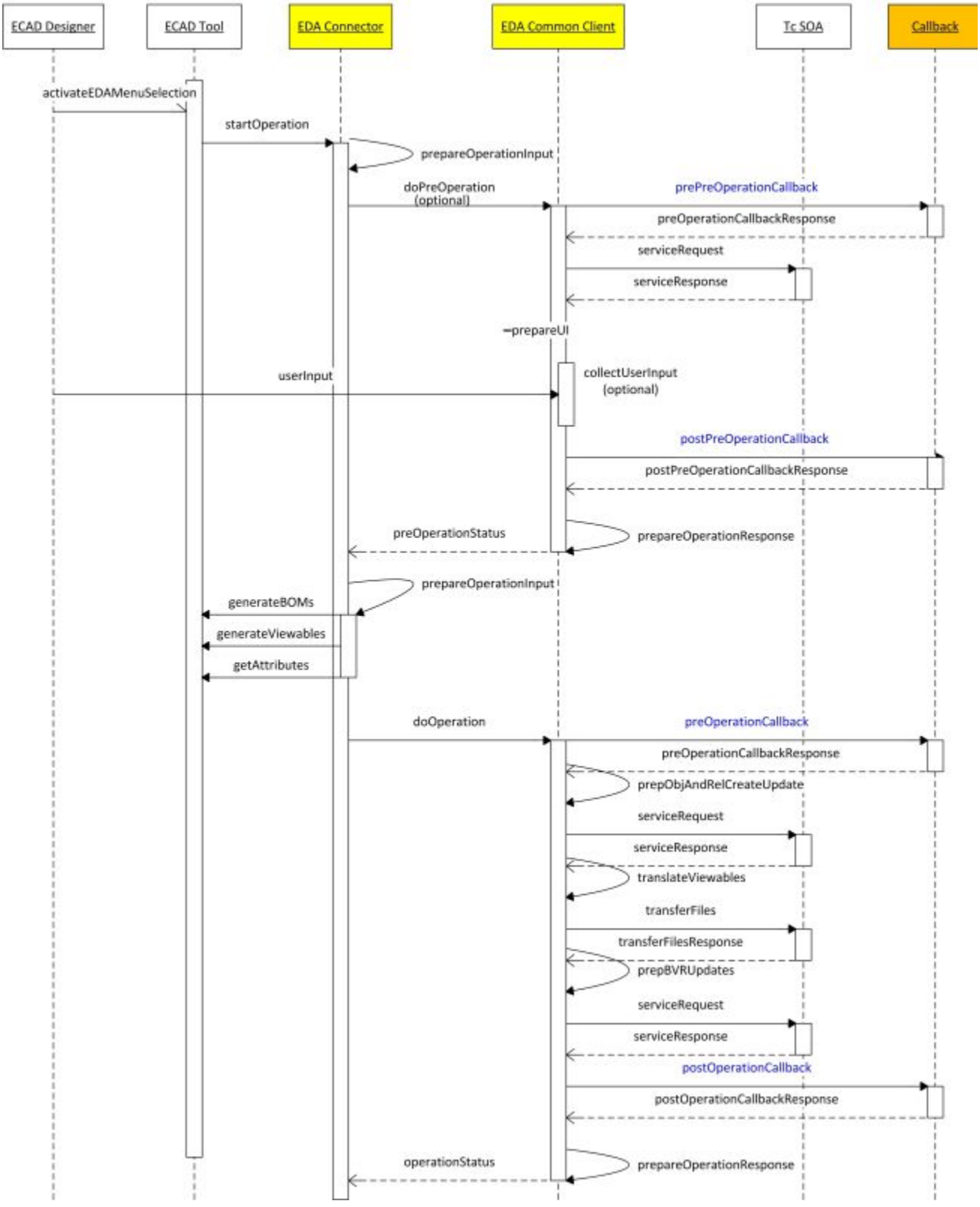
Some EDA common client operations have an associated **preOperation**. For example, the **Save As** operation must first collect metadata from the ECAD designer to identify the design being saved to Teamcenter and specify any additional information to be included in the save operation. This is accomplished by the **Save As preOperation**. When the ECAD designer specifies these details, the EDA connector extracts the additional required information from the ECAD design, such as, the Bills of Material and initiates the **Save As** operation. This operation then copies the ECAD design information into Teamcenter.

In general, all EDA common client operations and **preOperations** have two callbacks defined, **pre** and **post**. A pre operation callback is executed before the EDA common client operation or preOperation is performed. While a post operation callback is executed after the EDA common client operation or preOperation is performed.

For example, the **Save As** operation and associated **preOperation** each have pre- and post- callback named as follows:

- **prePreSaveAs** - The **Save As preOperation** pre callback
- **postPreSaveAs** - The **Save As preOperation** post callback
- **preSaveAs** - The **Save As Operation** pre callback
- **postSaveAs** - The **Save As Operation** post callback

The actions depicted in blue color in the below illutration shows the flow of pre and post **Operation** and **preOperation** callbacks for any EDA common client operation.



Apart from the generic pre and post operation callbacks, a few specialized callbacks are also available to perform customized EDA common client operations. These specialized callbacks are executed

at specific steps during the EDA common client operations. For example, a specialized callback **preDesignFileExtract** will be executed before the ECAD design files are downloaded and unzipped during the **Open** dataset operation.

> Note:
>
> This callback reference document provides supplementary information for the developers to use these callbacks to customize the EDA client operations only.

2. Writing a callback

To write a JAVA callback, perform following steps:

1. Select the callback that you want to use for the EDA operation.

   Refer to the callback reference section for **design** and **library** callbacks, which lists all callback methods available for implementation and the input parameters required for each callback.

2. Write a java class.

3. Add a method to implement the required behavior of the callback.

   The method includes the parameters that are passed to the callback. You can use these parameters to retrieve information about the EDA objects in the operations such as fetching the application name, EDADesign XML, EDAStatus XML, and so on.

   To retrieve information from EDA, you must use the classes and methods listed in the **EDA public classes and methods reference** section.

   > Note:
   >
   > As the selected method is the entry point for the callback implementation and it must be a public method.

4. Compile and build the callback implementation classes as explained in the **Building a callback** topic.

After adding your callback classes or scripts in EDA client environment, the EDA client must be configured to activate the callbacks for the pre/post operations they support.The callbacks must be configured in the EDA client definition XML as explained in the **Configuring callbacks** topic.

> Note:
>
> For a script callback, write a script (a batch file or a shell script) and provide the required implementation for the callback in this script. After writing the scritpt, configure the script as mentioned in the **Configuring callbacks** topic.

You can implement as many callbacks as required. Refer to an **example** to understand how to write a callback for the Cadence tool.

3. Example to write a callback for Cadence integration

Scenario: During a **Save As** operation for a schematic design in the Cadence ECAD tool, after the designer clicks **Save As**, an additional file must be also be created using the Cadence command and it must be uploaded along with the schematic design to Teamcenter as an attachment of a derived dataset.

To implement a **preSaveAs** java callback scenario:

1. Write a new java class, **PreSaveAsCallback.java**. It extends **java.lang.Object**.

2. Implement the method **createDerivedDataFile (String application, String edaDesignFileName)**.

   This method has the implementation to configure **datasetpath** in BMIDE while creating a derived data configuration. This path contains the location and the file name of the data file which will be stored as an attachment of the derived dataset.

3. When you perform a **Save As** operation for a Schematic in Cadence, the **Save As** dialog box is displayed.

   - In the first screen, you must provide details such as the **Item Id**, **ItemRevision Id**, and **Item Name**.

   - Select the **Generate Derived Data** option, to make sure that the derived data is created and saved along with the schematic design.

   - In the next screens, you must provide details for **CCABase**, **PWB**, and **CCA variants**.

     Provide details for the **Derived Item** and **Derived Dataset** that will be created in this operation.

   - Click **Finish**.

4. As result of this action, before the actual Save as operation is called in the EDA client, the **preSaveAs** callback configured for the schematic design in **cadenceSchematic_edadef.xml** is called.

   This callback is passed with the EDADesign parameters that are processed in the Save As operation and the application name, which is **cadenceSchematic** in this example.

5. The **createDerivedDataFile** method in the **PreSaveAsCallback** java class that implements the **preSaveAs** callback, fetches the location of the directory configured in BMIDE as the dataset path for the derived dataset.

6. The method also contains a call to a script that creates a data file, for example a PDF, using the Cadence command. The script is passed with the location of a directory configured for dataset path and it saves the created data file in this location.

   The name of the data file is the same as that configured in the BMIDE.

7. After the execution of this callback, the Save as operation is called. While creating and saving the main schematic design in Teamcenter, this operation also creates a derived dataset and picks up the data file created in the **preSaveAs** callback as an attachment and saves it with the derived dataset.

The **application** and **edaDesignFileName** parameters are passed to the **preSaveAs** callback when it is called from the EDA application.

You can use these parameters to retrieve the EDA design details and other related data from the EDA application. Refer to the **EDA public classes and methods reference** topic for more details.

After writing the callback class you must **build the java class** and include it in the EDA application.

4. Set property default values

The ECAD tool integration with Teamcenter EDA provides capability to set default property values for the properties of interest, which can be subsequently overridden by the designer on Teamcenter EDA common client user interface. The integration also provides capability to optionally mark the properties as read only on the EDA item creation user interface.

The **EDADesign** XML schema includes a **PropertyDefault** element to support an optional EDA object property value feature. EDA object property values may be specified to set the EDA logical item creation default values for any required or visible EDA logical item type properties on the corresponding EDA item creation user interface, and to optionally mark the properties as read only on the EDA item creation user interface. EDA object property values will be specified as an optional sequence of **PropertyType** sub-elements which can be inserted into an existing EDA Design XML file by the **prePreSaveAs**, **prePreRevise**, **prePreSave**, **prePreCheckin** callbacks for any new Items created while performing Save As, Save As copy, Revise, Save and Check-in operations.

The EDA object property default name specifies the target property for the associated value as a unique EDA logical type with optional variant name, Teamcenter type, property name and optional read only tag to make the EDA object property field non editable on Teamcenter EDA common client user interface. The default property value is specified as a string constant.

Refer to the EDADesign - PropertyDefault section for descriptions and examples in the *Teamcenter EDA Schema Reference* help on Support Center.

5. Building a callback

After writing a callback, build the callback as instructed below:

1. Organize the callbacks source files in a package structure as per your application.

   For example, java classes for the sample callbacks examples are in package, **com.teamcenter.edacallbacks** available at *%TCEDAECAD_ROOT%* **\example\Callback\java\com\teamcenter\edacallbacks** in the EDA install folder.

2. The *example* directory in the *%TCEDAECAD_ROOT%* folder contains callback examples. The **Callback\java** folder contains java source files and the **Callback\script** directory contains the callback script examples.

3. A sample build script, **build_callbacks.bat** for Windows or **build_callbacks.sh** for Unix, can be found under *%TCEDAECAD_ROOT%***\example\Callback\build** directory.

   The **build_callbacks.bat** in the **example/callbacks/build** folder provides a sample to show how the sample callback java package can be built. You must define the **JDK_HOME**, **TCEDAECAD_ROOT**, and **FMS_HOME** environment variables before you run the script.

4. Open the script in an editor. Read the instructions given in the script. Default values are provided for the environment variables in the script in the callback examples. Edit the script to specify values for the environment variables according to your environment, installation directories, and callback source code location.

5. Save the script.

6. Open command prompt for Windows environment or a UNIX Shell. Change to the directory where the **build_callbacks** script is present.

7. Run the script.

   A **CustomEDACallbacks.jar** file is created in the callbacks output location, *%TCEDAECAD_ROOT%* **\example\Callback**.

8. Copy this jar file to the *%TCEDAECAD_ROOT%* directory in your EDA client installation folder. This places the callback classes in the EDA classpath from where they are available for the EDA client use.

   If you change the name of the jar file from **CustomEDACallbacks.jar** then you must edit the **setup_eda** script in *%TCEDAECAD_ROOT%* directory and specify the custom jar file name in place of **CustomEDACallbacks.jar**.

6. Configuring callbacks

After adding your callback classes or scripts in the EDA client environment, the EDA client must be configured to activate the callbacks for the pre or post operations that the callbacks support.

To configure callbacks in the EDA client:

1. Check the schematic, simulation, or PCB design you are working on in the ECAD tool.

2. Depending on the ECAD application type, locate the EDA client definition XML file for the application in the *%TCEDAECAD_ROOT%* directory.

   For example, for Cadence, the XML file name will be **cadenceSchematic_edadef.xml** or **cadencePcb_edadef.xml**.

3. Edit the XML file to add the following configurations for the callbacks. The configuration elements must be added in a proper sequence. See the sequence and the XML syntax defined by the schema in **EDAClientDef.xsd**.

   For each of the JAVA callbacks:

   ```
   <CallbackDefs>
       <callback operation="<callback-operation-name>"
                 type="java"
                 command="<java-class-name-with-full-path>:<method-
   name>">
       </callback>
   </CallbackDefs>
   ```

   For each of the script callbacks:

   ```
   <CallbackDefs>
       <callback operation="<callback-operation-name>"
                 type="script"
                 command="<full-path-to-the-script>">
       </callback>
   </CallbackDefs>
   ```

   - The **command** attribute specifies the callback implementation class or script.
   - The **type** attribute is used to configure the type of callback as **java** or **script**.
   - The **operation** attribute specifies the callback name.

4. Save the XML.

5. Verify that there are no errors in the EDA client definition XML file to avoid the EDA client initialization to fail. You must check the XML file for syntax errors using Internet Explorer for Windows machine.

6. Restart the ECAD client application.

7. The callbacks are now available in the functionality.

An optional user configuration file named *<applicationName>*_**usrdef.xml** is available where the callbacks can be configured. The callbacks configured in this file are executed after the callbacks configured in the system configuration file, *<applicationName>*_**edadef.xml** are executed.

Example:

An example to configure callbacks for the Cadence client is given below. The configurations steps are explained for the **preSaveAs** JAVA callback explained in the **writing a callback example** topic. This callback is called in the Save As operation on a schematic design from the Cadence client.

The implementation class is **com.teamcenter.edabase.PreSaveAsCallback** and the implemented method used in the class is **createDerivedDataFile (String application, String edaDesignFileName)**.

1. Locate **cadenceSchematic_edadef.xml** in *%TCEDAECAD_ROOT%*.

2. Edit the XML file to add following lines inside the **EDAClientDef** XML element.

   For a java callback, the configuration appears as:

```
<CallbackDefs>
    <callback operation="preSaveAs"
              type="java"

command="com.teamcenter.edabase.PreSaveAsCallback:createDerivedDa
taFile">
    </callback>
</CallbackDefs>
```

   For a script callback, the configuration appears as:

```
<CallbackDefs>
    <callback operation="preSaveAs"
              type="script"
              command="d:\scripts\executePreSaveAs.bat">
```

```
            </callback>
        </CallbackDefs>
```

3. Verify that there are no errors in the modified EDA client definition XML file.

A sample configuration XML is provided along with the callbacks examples. The XML can be found in location *%TCEDAECAD_ROOT%***\example\config\exampleSchematic_edadef.xml**.

7. EDA public classes and methods reference

A list of java classes and methods for EDA and the details on how to use these classes and methods are given below. You can use these to access objects in the EDA application while writing callbacks.

DefaultSOAService

This class must to be used to get the SOA connection.

```
import com.teamcenter.edabase.services.DefaultSOAService;
import com.teamcenter.soa.client.Connection;

DefaultSOAService service = new DefaultSOAService().
Connection conn = service.getConnection().
```

The returned SOA Connection object **conn**, can be used further to call required SOA services.

TcEDALogger

This class is no longer used for normal logging purposes. You can use **log4j** to add log statements to the EDA log file, as explained below:

```
import org.apache.log4j.Logger;
private static final Logger s_logger =
Logger.getLogger( EDAConfiguration.class );
s_logger.debug( "This is a debug message" );
s_logger.error( "This is a normal error" );
s_logger.fatal( "This is a fatal error" );
s_logger.info(  "This is a informative message" );
s_logger.warn(  "This is a warning" );
```

EDAConfiguration

Use **methodgetStagingDirName()** from this class to fetch the staging directory for EDA designs in the operation. When a design is downloaded from Teamcenter by an ECAD tool, the design is saved in this directory.

You can fetch the path of this directory programmatically in your callbacks using this method.

```
import com.teamcenter.edabase.EDAConfiguration;
```

```
String stagingDirName =
EDAConfiguration.getInstance().getStagingDirName()
```

EDAException

EDAException object can be used to handle exception scenarios in your callbacks. When you use this object to handle exceptions in error cases or in exception cases, EDA recognizes those exceptions when the control returns back to EDA from your callbacks. This way the callbacks are executed in the flow of the EDA operations.

When **EDAException** is generated or is thrown from a callback, it is logged in EDA and an error message is displayed on the EDA client. However, the EDA operation continues to execute as is.

EDAException can be used as explained below:

```
import com.teamcenter.edabase.EDAException;
throw new EDAException( "Item revision not found" );
```

Refer to the example callback **PostOpenCallback.java** in the %TCEDAECAD_ROOT% **\example\Callback\java\com\teamcenter\edacallbacks** folder for more details.

EDACancelException

EDACancelException is an EDA exception handling object that cancels an EDA operation that is being executed. You can use this object in certain scenarios in your callbacks, where you want to stop the EDA operation from proceeding further.

When you throw **EDACancelException** from your callback, it is logged in EDA. The error message you included in the exception is displayed on the EDA client and the exception is thrown back in EDA so that the EDA operation you performed is aborted.

> Note:
>
> Canceling an EDA operation in this way is possible only from the pre operation callbacks. For post operation callbacks, the EDA operation is already executed. These callbacks are essentially provided to process results of the EDA operation they are executed for.

For example, in the example callback – **PreSaveCallback**, when you perform a Save operation on a schematic design in the Cadence client, EDA performs a status check to verify whether you have selected the flag **WithBOM** as true or not. The callback throws **EDACancelException**. This exception is caught in the EDA error message **Design variants set without BOM.**

A message (**Save cancelled.**) is displayed and the exception is thrown back, which in turn ensures that EDA save operation does not continue.

You can use **EDACancelException** as explained below:

```
import com.teamcenter.edabase.EDACancelException;

throw new EDACancelException( "Design variants set without BOM. Save
cancelled." );
```

UIUtils

This class contains the following methods which may be used to display simple dialogs.

```
showConfirmMessageBox(String, String)
showErrorMessageBox(String, String)
showErrorMessageBox(String, String, List<Status>)
showInformationMessageBox(String, String)
showQuestionMessageBox(String, String)
showWarningMessageBox(String, String)
```

Example

```
import com.teamcenter.edabase.utils.UIUtils;

boolean confirm = UIUtils.showConfirmMessageBox( "Confirm",
        "Do you want to continue?" );
If( !confirm )
{
    UIUtils.showErrorMessageBox( "Error",
            "User has elected to cancel the operation" );
}
```

8. EDA design integration callback reference

EDA design integration callback summary

In an EDA operation, to call a callback for a specific type of application, that is, for PCB, schematic, or simulation, configuration must to be done in the corresponding EDA client definition XML file.

For example, for the Cadence client, to include callbacks for Schematic, the callbacks must be configured in the EDA client definition XML file, **cadenceSchematic_edadef.xml**. Refer to the **configuring callbacks** topic for details on configuring callbacks for the Cadence client.

The table below lists commonly used callbacks supported by EDA for embedded connectors for your quick reference. The first column contains the name of callbacks that must be specified in the configuration as the operation name. The second column briefly describes when and the callback is called by the EDA client and how it should work.

Callback	Description
preOpen	An embedded callback that executes before open operation.
postOpen	An embedded callback that executes after open operation.
prePreSave	An embedded callback that executes before save operation. This callback will be called before the save dialog appears.
postPreSave	An embedded callback that executes before save operation. This callback will be called after the user clicks finish, but before control is returned to the connector.
preSave	An embedded callback that executes before save operation. After the user clicks finish, and the connector has called –presave, this callback is called.
postSave	An embedded callback that executes after save operation, before returning control to the connector.
prePreSaveAs	An embedded callback that executes before save as operation. This callback will be called before the save as dialog appears.
postPreSaveAs	An embedded callback that executes before save as operation. This callback will be called after the user clicks finish, but before control is returned to the connector.
preSaveAs	An embedded callback that executes before save as operation. After the user clicks finish, and the connector has called –presaveas, this callback is called.
postSaveAs	An embedded callback that executes after save as operation, before returning control to the connector.

Callback	Description
prePreRevise	An embedded callback that executes before the Revise dialog is displayed to collect user input for the Revise operation. This callback processes EDADesign XML information prior to display of the Revise dialog.
postPreRevise	An embedded callback that executes after user input has been collected for the Revise operation and the user clicks finish. This callback processes EDADesign XML and EDAStatus XML information.
preRevise	An embedded callback that executes before the Teamcenter Revise operation. This callback processes the EDADesign XML prior to revising the design in Teamcenter.
postRevise	An embedded callback that executes after design is revised in Teamcenter and processes the EDADesign XML and EDAStatus XML resulting from revising the design in Teamcenter.
preCheckout	An embedded callback that executes before check out operation.
postCheckout	An embedded callback that executes after check out operation.
prePreCheckin	An embedded callback that executes before the Check-In dialog is displayed to collect user input for the Check-In operation. This callback processes EDADesign XML information prior to display of the Check-In dialog.
postPreCheckin	An embedded callback that executes after user input has been collected for the Check-In operation and the user click finish. This callback processes EDADesign XML and EDAStatus XML information.
preCheckin	An embedded callback that executes before Checkin operation.
postCheckin	An embedded callback that executes after Checkin operation.
preCancelCheckout	An embedded callback that executes before Cancel-Checkout operation.
postCancelCheckout	An embedded callback that executes after Cancel-Checkout operation.
preRefresh	An embedded callback that executes before Refresh operation.
postRefresh	An embedded callback that executes after Refresh operation.
preSaveDerivedData	An embedded callback that executes before Save Derived Data operation.
postSaveDerivedData	An embedded callback that executes after Save Derived Data operation.
preGetItemInfo	An embedded callback that executes before Get Item Info operation.
postGetItemInfo	An embedded callback that executes after Get Item Info operation.
preShowItemInfo	An embedded callback that executes before the Show Design Info, and deprecated Show Item Info, operations.
postShowItemInfo	An embedded callback that executes after the Show Design Info, and deprecated Show Item Info, operations .
postPurgeCache	An embedded callback that executes after Purge Cache operation.
prePDMOpen	An embedded callback that executes before open operation called from PDM system.

Callback	Description
postPDMOpen	An embedded callback that executes after open operation called from PDM system.
preDownloadBOM	An embedded callback that executes before Download BOM operation.
postDownloadBOM	An embedded callback that executes after Download BOM operation.
preDesignFileExtract	A specialized callback that executes before download and unzipping of the design files on open operation. The purpose of this new callback is to manage design files that may need to be restored or merged following download and unzipping of the design files. One example where this callback would be used is to backup the existing design files before the open operation downloads the design files and overwrite the existing design files.
nameDerivedItem	A specialized callback that is executed on Save As, Save, Check-in and Revise operations while naming Derived Item on Derived Item Table page. This callback is executed once for each Derived Item row on the Derived Item Table page. This callback returns a string which is used as the default value for Derived Item name in place of EDA Client composed default value. It can be overridden in the UI by the user. If this callback returns null or an empty string then the Derived Item name will be defaulted to the EDA Client composed default value.
nameDerivedDataset	A specialized callback that is executed on Save As, Save, Check-in, Revise and Save Derived Data operations while naming Derived Dataset on Derived Dataset Table page. This callback is executed once for each Derived Dataset row on the Derived Dataset Table page. This callback returns a string which is used as the default value for Derived Dataset name in place of EDA Client composed default value. It can be overridden in the UI by the user. If this callback returns null or an empty string then the Derived Dataset name will be defaulted to the EDA Client composed default value.
validateEDAComponents	A specialized callback that is executed during Save As, Save, Check-in, Revise, and BOM Compare operations. This callback is executed once for all ECAD components specified in the BOM(s) generated for the operations listed above. This callback provides for customized validation of each ECAD component, based on the component Item ID specified in the ECAD BOM(s), to determine if it is a Teamcenter electrical component that can be used in ECAD designs. This callback overrides COTS EDA Client validation of ECAD components by item types (see EDA_ComponentItemTypes preference).

A summary of all the optional Java embedded callback methods and the **BMIDE configured derived dataset callback** method for EDA design integration supported by the EDA common client for customization purposes is listed in the next section. Each summary provides the method signature, brief argument descriptions, and the return value. Custom method implementation must conform to the specified method signature.

Refer to the EDADesign and EDAStatus for descriptions of the design and status files mentioned in these callbacks in the Schema Reference help on Support Center.

preOpen

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - The name of the operation design file to be populated with information about the design. If this file exists, the contents should be ignored at this time. This file is created/populated by a successful Open operation.

Returns:

The method returns void.

postOpen

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design which was opened including, but not limited to, the UID of the primary design dataset.

statusFileName - A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if

the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum.

Returns:

The method returns void.

prePreSave

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design required to initiate a Save operation including, but not limited to, the UID of the primary design dataset.

The **prePreSave** callback has an ability to optionally set default values for the properties of interest and to optionally mark the properties as read only on the EDA item creation UI. Refer to the **Setting Property Default Values** section for more details.

Returns:

The method returns void.

postPreSave

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains BOM and Viewable options specified by designer as well as attributes to be extracted from the ECAD design.

Returns:

The method returns void.

preSave

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset. For this callback, the design file also contains BOM and viewable information.

Returns:

The method returns void.

postSave

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the newly created primary dataset version.

Returns:

The method returns void.

prePreSaveAs

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design required to initiate a SaveAs operation including, but not limited to, the UID of the primary design dataset. For this callback, this file also contains the folder for the design.

The **prePreSaveAs** callback has an ability to optionally set default values for the properties of interest and to optionally mark the properties as read only on the EDA item creation UI. Refer to the **Setting Property Default Values** section for more details.

Returns:

The method returns void.

postPreSaveAs

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset. For this callback, this file also contains the folder for the design.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the Item ID, Item Name, and BOM and Viewable options specified by the designer; the new design folder, saveAsCopy indicator, and attributes to be extracted from the ECAD design.

Returns:

The method returns void.

preSaveAs

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset. For this callback, the design file also contains BOM and viewable information.

Returns:

The method returns void.

postSaveAs

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset.

statusFilename - A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the newly created primary dataset version, the saveAsCopy indicator, and checkIn indicator.

Returns:

The method returns void.

prePreRevise

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design required to initiate a Revise operation including, but not limited to, the UID of the primary design dataset. For this callback, this file also contains the folder for the design.

The **prePreRevise** callback has an ability to optionally set default values for the properties of interest and to optionally mark the properties as read only on the EDA item creation UI. Refer to the **Setting Property Default Values** section for more details.

Returns:

The method returns void.

postPreRevise

Signature:

```
public void methodName(    java.lang.String applicationName,
```

```
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Revise operation including, but not limited to, the UID of the primary design dataset. For this callback, this file also contains the folder for the design.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the Item ID, Item Name, and BOM and Viewable options specified by the designer; the new design folder, Revise indicator, and attributes to be extracted from the ECAD design

Returns:

The method returns void.

preRevise

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Revise operation including, but not limited to, the UID of the primary design dataset. For this callback, the design file also contains BOM and viewable information.

Returns:

The method returns void.

postRevise

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Revise operation including, but not limited to, the UID of the primary design dataset.

statusFilename - A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the newly created primary dataset version, the Revise indicator, and checkIn indicator.

Returns:

The method returns void.

preCheckout

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file to be created by the checkout operation.

uid – A string containing the UID of the dataset to be checked out.

Returns:

The method returns void.

postCheckout

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design which was refreshed including, but not limited to the UID and location of the primary design dataset and already open related datasets. If the primary and already opened related datasets as specified by the client configuration are already the latest version, no design downloads will occur and this file will not be populated.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the latest primary dataset version and refresh indicators for the primary and related designs as defined by the client configuration.

uid – A string containing the UID of the dataset that was checked out.

Returns:

The method returns void.

prePreCheckin

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design specified for check in, including, but not limited to, the UID of the primary design dataset.

The **prePreCheckin** callback has an ability to optionally set default values for the properties of interest and to optionally mark the properties as read only on the EDA item creation UI. Refer to the **Setting Property Default Values** section for more details.

Returns:

The method returns void.

postPreCheckin

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design specified for check in, including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the newly created primary dataset version.

Returns:

The method returns void.

preCheckin

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design including, but not limited to, the UID of the primary design dataset.

Returns:

The method returns void.

postCheckin

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design which was checked in including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains the UID of the newly created primary dataset version.

Returns:

The method returns void.

preCancelCheckout

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - The name of the operation design file to be populated with information about the design. If this file exists, the contents should be ignored at this time. This file is created/populated by a successful un-Checkout operation.

uid – A string containing the UID of the dataset to be un-checked out.

Returns:

The method returns void.

postCancelCheckout

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design which was un-checked out including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum.

uid – A string containing the UID of the latest design dataset version.

Returns:

The method returns void.

preRefresh

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file to be created by the refresh operation.

uid – A string containing the UID of the primary dataset of the design to be refreshed.

Returns:

The method returns void.

postRefresh

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design which was refreshed including, but not limited to the UID and location of the primary design dataset and already open related datasets. If the primary and already

opened related datasets as specified by the client configuration are already the latest version, no design downloads will occur and this file will not be populated.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the EDA Design XML file should be ignored at a minimum.

For this callback, the status file also contains refreshed indicator and resulting UID of the primary and related datasets.

uid – A string containing the UID of the primary dataset of the design to be refreshed.

Returns:

The method returns void.

preSaveDerivedData

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design for which derived data is to be saved including, but not limited to, the UID of the primary design dataset.

Returns:

The method returns void.

postSaveDerivedData

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design for which derived data was to be saved including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

Returns:

The method returns void.

preGetItemInfo

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

uid – A string containing the UID of the dataset of interest.

Returns:

The method returns void.

postGetItemInfo

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design of interest including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

uid – A string containing the UID of the dataset of interest.

Returns:

The method returns void.

preShowItemInfo

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String uid,
    java.lang.String extraInfo )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

uid – A string containing the UID of the dataset of interest.

extraInfo – A string containing a comma-separated list of colon-delimited name value pairs to be displayed. For example, **Username:Tester**,**OS:Windows XP**.

Returns:

The method returns void.

postShowItemInfo

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid,
    java.lang.String extraInfo )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - The name of the operation design file. This file is an XML file which contains the design details of the EDA design which is being worked on in the operation. It contains values like UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

uid – A string containing the UID of the dataset of interest.

extraInfo – A string containing a comma-separated list of colon-delimited name value pairs to be displayed. Example: Username:Tester,OS:Windows XP.

Returns:

The method returns void.

postPurgeCache

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

Returns:

The method returns void.

prePDMOpen

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String datasetID )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the operation design file to be populated with information about the design. If this file exists, the contents should be ignored at this time. This file is created/populated by a successful Open operation.

datasetID - A string containing the UID of the dataset to be opened.

Returns:

The method returns void.

postPDMOpen

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String datasetID )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the operation design file. This XML file contains the design details of the EDA design which was opened including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

datasetID - A string containing the UID of the dataset that was opened.

Returns:

The method returns void.

preDownloadBOM

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the operation design file to be populated with information about the design. If this file exists, the contents should be ignored at this time. This file is created/populated by a successful download BOM operation.

uid – A string containing the UID of the dataset for BOM download.

Returns:

The method returns void.

postDownloadBOM

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName,
    java.lang.String uid )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the operation design file. This XML file contains the design details of the EDA design which was downloaded including, but not limited to, the UID of the primary design dataset and BOM components information.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

uid – A string containing the UID of the dataset that was downloaded.

Returns:

The method returns void.

preDesignFileExtract

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA designs which are being opened including, but not limited to, the UID and the source location of the design data. This XML file will only contain dataset elements for the downloaded designs that will be extracted. For instance, if the schematic design has already been downloaded and the designer opens the related simulation design, in this case, the XML file will only contain a simulation Dataset element. This XML file will not contain the checkout status of the downloaded datasets.

Returns:

The method returns void.

preInitiateIssueManager

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - The name of the operation design file to be populated with information about the design. The content mainly contains two aspects as listed below:

- **UID** – design UID which will be processed to attached to new issue report object via problem item relation.

- **Dataset** – A list of dataset under CCA element. If these dataset types match the dataset type defined in IssueAttachmentsDataType in EDA client def file, system will automatically create/update the dataset object with file attachments.

  Other elements – System ignores other elements in eda design xml if it contains, for example, component.

This file is created/populated by a successful **initiateIssueManager** operation.

Returns:

The method returns void.

postInitiateIssueManager

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA design file. This XML file contains the design details of the EDA design which was generated by preInitiateIssueManager and initiateIssueManager operation.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum.

Returns:

The method returns void.

preSaveDerivedData2

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design for which derived data is to be saved including, but not limited to, the UID of the primary design dataset.

Returns:

The method returns void.

postSaveDerivedData2

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName,
    java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains the design details of the EDA design for which derived data was to be saved including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution.

Returns:

The method returns void.

prePreSaveDerivedData2

Signature:

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaDesignFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName - A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design required to initiate a Save operation including, but not limited to, the UID of the primary design dataset.

The **prePreSaveDerivedData2** callback has an ability to optionally set default values for the properties of interest and to optionally mark the properties as read only on the EDA item creation UI. Refer to the Setting Property Default Values section for more details.

Returns:

The method returns void.

postPreSaveDerivedData2

Signature:

```
public void methodName(    java.lang.String applicationName,
```

```
java.lang.String edaDesignFileName,
java.lang.String statusFileName )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignFileName – A string specifying the name of the EDA operation design file. This XML file contains design details of the ECAD design for the Save operation including, but not limited to, the UID of the primary design dataset.

statusFileName – A string specifying the name of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaDesignFile should be ignored at a minimum. For this callback, the status file also contains BOM and Viewable options specified by designer as well as attributes to be extracted from the ECAD design.

Returns:

The method returns void.

BMIDE configured derived dataset callback

Signature:

```
public void methodName(java.lang.String edaDesignFileName)
```

Argument:

edaDesignFileName - A string specifying the name of the EDA design file. This XML file contains details of the ECAD design submitted for the operation including, but not limited to, the UID of the primary design dataset and the folder containing the design.

For variant derived data callbacks, the first CCAVariant sub-element of the CCA element is the variant of interest for the callback.

Returns:

The method returns void.

nameDerivedItem

Signature:

```
public String methodName(java.lang.String applicationName,
    DerivedItemNamingInfo derivedItemNamingInfo )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

derivedItemNamingInfo - **DerivedItemNamingInfo** class object that contains information regarding existing derived items, derived item configuration, schematic, CCA, and PWB items that is required to construct new derived item name. The **nameDerivedItem** callback can make use of the following accessor methods from **DerivedItemNamingInfo** class object while constructing derived item name.

```
public String getParentItemID()
    Returns Derived Item’s parent Item ID

public String getDefaultDerivedItemName()
    Returns default Derived Item name constructed by EDA

public String[] getSiblingDerivedItemNames()
    Returns the names of other Derived Items sharing the same parent

public String getDerivedItemConfiguredName()
    Returns configured Derived Item name

public String getDerivedItemConfiguredPrefix()
    Returns configured Derived Item prefix

public String getDerivedItemConfiguredSuffix()
    Returns configured Derived Item suffix

public String getCCAItemID()
    Returns CCA/CCABase Item ID

public String getSchematicItemID()
    Returns Schematic Item ID

public String getPWBItemID()
    Returns PWB Item ID
```

Returns:

The method returns a string containing derived item name constructed in this callback, null otherwise.

nameDerivedDataset

Signature:

```
public java.lang.String methodName(
    java.lang.String applicationName,
    DerivedDatasetNamingInfo derivedDatasetNamingInfo )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

derivedDatasetNamingInfo – **DerivedDatasetNamingInfo** class object that contains information regarding existing derived datasets, derived dataset configuration, Schematic, CCA, and PWB items that is required to construct new derived dataset name. The **nameDerivedDataset** callback can make use of the following accessor methods from **DerivedDatasetNamingInfo** class object while constructing derived dataset name.

```
public String getParentItemID()
    Returns Derived Dataset's parent Item ID

public String getDefaultDerivedDatasetName()
    Returns default Derived Dataset name constructed by EDA

public String[] getSiblingDerivedDatasetNames()
    Returns the names of other Derived Datasets sharing the same parent

public String getDerivedDatasetConfiguredName()
    Returns configured Derived Dataset name

public String getDerivedDatasetConfiguredPrefix()
    Returns configured Derived Dataset prefix

public String getDerivedDatasetConfiguredSuffix()
    Returns configured Derived Dataset suffix

public String getCCAItemID()
    Returns CCA/CCABase Item ID
```

```
public String getSchematicItemID()
    Returns Schematic Item ID

public String getPWBItemID()
    Returns PWB Item ID
```

Returns:

The method returns a string containing derived dataset name constructed in this callback, null otherwise.

validateCADComponents

Signature:

```
public ECADComponentValidationInfo methodName(
    java.lang.String applicationName,
    ECADComponentValidationInfo ecadComponentValidationInfo )
```

Argument:

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

ecadComponentValidationInfo – **ECADComponentValidationInfo** class object that contains the candidates for all ECAD components specified in the operation BOM(s) for validation. The following accessor methods are provided by this class for use by the **validateECADComponents** callback for validating the candidates for each ECAD component.

```
public String getFirstComponentItemIdToValidate()
   Establishes the first ECAD component as the current ECAD component
for validation and returns the corresponding Item ID.

public String getNextComponentItemIdToValidate()
   Sets the next ECAD component as the current ECAD component for
validation and returns the corresponding Item ID. Subsequent calls to
this method return subsequent ECAD components. When all ECAD components
have been processed, null is returned.

public String getFirstComponentCandidateUIDToValidate()
   Establishes the first candidate for the current ECAD component for
validation and returns the corresponding candidate UID. Null is returned
```

```
in the case there are no candidates for the ECAD component.

public String getNextComponentCandidateUIDToValidate()
   Steps to the next candidate for the current ECAD component for
validation and returns the corresponding candidate UID. Subsequent calls
to this method return subsequent ECAD component candidates. When all
candidates have been processed for the current ECAD component, null is
returned.

public ModelObject getComponentCandidateObj( String uid )
   Returns the candidate ModelObject for validation corresponding to the
specified UID.

public void recordValidatedComponentCandidate( String uid )
   Records the specified candidate that was validated for the current
ECAD component according to the requirements implemented by this
callback.
```

Returns:

The method returns void.

custom_<operationName>

Signature:

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

Argument:

operationName – A string specifying the operation name for the generic API

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **cadenceSchematic** in case of a Cadence Schematic EDA client.

edaDesignPathName – The path of the operation design file to be populated with information about the design. The callbacks should use it to communicate design info between the tool (connector) and EDA common Client. This is an optional API input. The callbacks should ignore it when the generic custom_<operationName> API is not called with this argument.

statusPathName - A string specifying the path of the EDA operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution such as cancel or error handling. This is an optional API input. The callbacks should ignore it when the generic custom_<operationName> API is not called with this argument.

extraArgs – An array of strings including additional arguments passed from the generic API call. This is an optional API input. The callbacks should check whether it is null/empty or not before use it.

Returns:

The method returns void.

9. EDA library integration callback reference

EDA library integration callback summary

A list of callbacks available in the EDA client that can be implemented and called for corresponding EDA library operation is given below.

Callback	Description
preLoadLibrary	An embedded callback that executes before loading library components from Teamcenter to ECAD library, while performing Load Library operation.
postLoadLibrary	An embedded callback that executes after loading library components from Teamcenter to ECAD library, while performing Load Library operation.
preExportLibrary	An embedded callback that executes before exporting library components from ECAD library to Teamcenter, while performing Export Library operation.
postExportLibrary	An embedded callback that executes after exporting library components from ECAD library to Teamcenter, while performing Export Library operation.
preSyncLibrary	An embedded callback that executes before synchronizing library components from Teamcenter to ECAD library, while performing Sync Library operation.
postSyncLibrary	An embedded callback that executes after synchronizing library components from Teamcenter to ECAD library, while performing Sync Library operation.
preSaveLibrary	An embedded callback that executes before saving library components from ECAD library to Teamcenter, while performing Save Library operation.
postSaveLibrary	An embedded callback that executes after saving library components from ECAD library to Teamcenter, while performing Save Library operation.
preReviseLibrary	An embedded callback that executes before revising library components in Teamcenter, while performing Revise Library operation.
postReviseLibrary	An embedded callback that executes after revising library components in Teamcenter, while performing Revise Library operation.
preCheckoutLibrary	An embedded callback that executes before check-out of library components in Teamcenter, while performing Check-Out Library operation.
postCheckoutLibrary	An embedded callback that executes after check-out of library components in Teamcenter, while performing Check-Out Library operation.
preCheckinLibrary	An embedded callback that executes before check-in of library components in Teamcenter, while performing Check-In Library operation.

Callback	Description
postCheckinLibrary	An embedded callback that executes after check-in of library components in Teamcenter, while performing Check-In Library operation.
preCancelCheckoutLibrary	An embedded callback that executes before cancel check-out of library components in Teamcenter, while performing Cancel Check-Out Library operation.
postCancelCheckoutLibrary	An embedded callback that executes after cancel check-out of library components in Teamcenter, while performing Cancel Check-Out Library operation.

A summary of all the optional Java embedded callback methods for EDA library integration supported by EDA common client for customization purposes are explained separately. Each summary provides the method signature, brief argument descriptions, and return value. Custom method implementation must conform to the specified method signature.

You can also Refer to the EDALib and EDAStatus for descriptions of the library and status files mentioned in these callbacks in the *Schema Reference* help on Support Center.

preLoadLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaLibXmlFileName )
```

Arguments

applicationName – A string specifying the application name that is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file

Returns

The method returns void.

postLoadLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String edaLibXmlFileName,
                              java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preExportLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaLibFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibFileName - A string specifying the name of the EDA part library XML or JSON file (This file can be JSON file, if JSON file is passed as a command line input). This file contains information pertaining to EDA library parts.

Returns

The method returns void.

postExportLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String edaLibFileName,
                              java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibFileName - A string specifying the name of the EDA part library XML or JSON file (This file can be JSON file, if JSON file is passed as a command line input). This file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preSyncLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String edaLibFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibFileName - A string specifying the name of the EDA part library XML or JSON file (This file can be JSON file, if JSON file is passed as a command line input). This file contains information pertaining to EDA library parts.

Returns

The method returns void.

postSyncLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String edaLibFileName,
                              java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

edaLibFileName - A string specifying the name of the EDA part library XML or JSON file (This file can be JSON file, if JSON file is passed as a command line input). This file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLiblFileName should be ignored at a minimum.

Returns

The method returns void.

preSaveLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String selectedObjectXmlFileName,
                           java.lang.String edaLibXmlFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to save.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file.

Returns

The method returns void.

postSaveLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String selectedObjectXmlFileName,
    java.lang.String edaLibXmlFileName,
                           java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to save.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preReviseLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String selectedObjectXmlFileName)
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to revise.

Returns

The method returns void.

postReviseLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String selectedObjectXmlFileName,
                    java.lang.String edaLibXmlFileName,
                           java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to revise.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preCheckoutLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String selectedObjectXmlFileName)
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to checkout.

Returns

The method returns void.

postCheckoutLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String selectedObjectXmlFileName,
                        java.lang.String edaLibXmlFileName,
                           java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to checkout.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preCheckinLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String selectedObjectXmlFileName,
                           java.lang.String edaLibXmlFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to check-in.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file.

Returns

The method returns void.

postCheckinLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String selectedObjectXmlFileName,
    java.lang.String edaLibXmlFileName,
                           java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to check-in.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.

preCancelCheckoutLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
    java.lang.String selectedObjectXmlFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to cancel check out.

Returns

The method returns void.

postCancelCheckoutLibrary

Signature

```
public void methodName(    java.lang.String applicationName,
                           java.lang.String selectedObjectXmlFileName,
    java.lang.String edaLibXmlFileName,
                           java.lang.String statusFileName )
```

Arguments

applicationName – A string specifying the application name which is executing this callback. This is the value of the application attribute specified in the EDA Client configuration.

For example, **mentorExpLib** in case of a Mentor Xpedition library integration.

selectedObjectXmlFileName - A string specifying the name of the EDA Part Library XML file which holds Item IDs user selected on the Gateway main UI to cancel check out.

edaLibXmlFileName - A string specifying the name of the EDA Part Library XML file. This XML file contains information pertaining to EDA library parts.

statusFileName - A string specifying the name of the EDA library operation status file. This XML file contains the operation status. Callbacks should use the status to control the callback execution. For instance, if the status indicates that the operation was canceled or encountered an error, the edaLibXmlFile should be ignored at a minimum.

Returns

The method returns void.