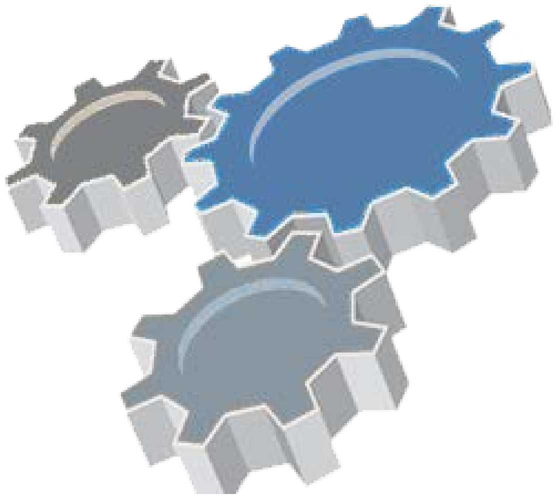
Teamcenter Integration for CATIA V5 (12.0.1) Installation Guide



June, 2018

Legal Notice

Table of Contents

1.0 What's New in 12.0.1

1.1 New Features and Improvements

System Requirements

Visualization in Active Workspace

1.1.1 System Requirements

In addition to other CATIA/CAA Compatibilities, Teamcenter Integration for CATIA V5 (12.0.1) also supports CATIA V5-6R2017 and CATIA V5-6R2018.

1.1.2 Visualization in Active Workspace

Visualization is supported with Active Workspace 4.0 by activating the IS_VIEWER_SUPPORTED option in the *%TCICV5_DIR%\env\aw_config.properties* file.

Set this option to **false** to disable visualization in Active Workspace.

```
awhosted.option.IS_VIEWER_SUPPORTED = true
```

1.2 Known Issues

1.2.1 Documentation Issue

Links in the search results are disabled when the **%** character is used in the search box.

Workaround

Omit the % character when searching. For example, enter *WORK_DIR* instead of *%WORK_DIR%*.

1.3 Closed Problem Reports

Function		PR Number		Description
Export		3056279		Export drawing with vew from 3D option gives error though export is successful
Import		8903397		TcIC Spreadsheet Import for Drawing with MFK counter fails for second MFK value
Load		8538434		When load assembly, structure is not sync
Load		8939585		Can not open CGR File
Load		8965257		TcIC - load to CATIA revision conflict last column not visible
Load for consulting		3056499		Load for consulting & compare shows TcIC save manager when trying to save.
Purge		8976922		Purge deletes new files in CATIA Session when DLNames is used.
Replace		8979483		Replace loading as CGR leads to catia hang.
Save		3056116		Save as with Scene management increases save time considerably
Save		8981066		TcIC does not handle multi-variable DL_Names
Save		9011182		CATIA TcIC: Scene (arrangement) do not display correctly in Viewer.
Save		9133166		TcIC occasionally hangs when running in silent mode
Save As		9050628		TcIC Error when store CATProcess dataset to existing ICF_Cust_Prt-item.
Save Manager		8933438		LOV type property is not available in save manager
Save Manager		9058058		Save Manager Multi-Select Use Existing Dataset gets Null Error
State Details		8955167		TcIC is use a lot of memory

2.0 Installation Prerequisites

Ensure the following requirements have been met before using Teamcenter Integration for CATIA V5.

Supported Operating Systems
System Requirements
CAA Compilation Prerequisites
Shared Library Clean-up
Localization / Internationalization

***Templates** must also be configured before using Teamcenter Integration for CATIA V5.*

*The MCAD Integration template is required when using Embedded **Active Workspace**.*

2.1 Supported Operating Systems

Refer to the Siemens PLM Hardware and Software Certifications page for OS **server** side support.

SIEMENS	MS
Active Workspace 3.1*	Windows 7 Windows 8 Vista Windows 10
Teamcenter 10.1.5 (Client)*	Windows 7 Windows 8 Vista
Teamcenter 10.1.6 (Client)*	Windows 7 Windows 8 Windows 10
Teamcenter 11.3 (Client)*	Windows 7 Windows 8 Windows 10

**and higher in the same series.*

2.2 System Requirements

2.2.1 Teamcenter and CATIA Requirements

- Teamcenter 10.1.5 or 11.3 and higher in the same series for UNIX and Windows platforms. Active Workspace 3.1 and higher. Refer to Supported Operating Systems for specific details.
- CATIA V5R18 to V5-6R2018 and corresponding CAA.
- Set up licenses for all the products.

2.2.2 CATIA/CAA Compatibilities

The following table lists compatibilities between the CATIA/CAA and the minimum level supported to the Integration. Refer to Dassault Systèmes for specific OS support.

CATIA V5 / CAA V5	Service Pack
R18	SP4
R19	SP3 minimum
R20	SP1
R21	SP1
V5-6R2012, V5-6R2013, V5-6R2014, V5-6R2015	SP1
V5-6R2016, V5-6R2017	SP1, SP2

CATIA V5 / CAA V5	Service Pack
V5-6R2018	SP1 minimum

2.2.3 JAVA Version

Beginning with version 4.1.1 of Teamcenter Integration for CATIA V5, the required JAVA version is the one used by Teamcenter.

2.2.4 HP-UX 11.11 Prerequisite

To run the Teamcenter Integration for CATIA V5 on HP-UX 11.11, the Kernel parameters must be set to:

- MSGSEG=32767
- NFLOKS= 4096
- SHMINI= 512

2.3 CAA Compilation Prerequisites

Ensure the following prerequisites have been met on the client machine (Windows, AIX or SUN) for successful CAA compilation.

- Teamcenter Client (2-Tier or 4-Tier)
- CATIA V5 Release xx
- CAA API CATIA V5 Release xx
- CAA RADE (Visual Studio or other) for CATIA V5 Release xx
- CAA RADE License: ABC, CDV or CDC
- C/C++ Compiler
- Visual Studio

Refer to System Requirements for the minimum version of CAA which is supported by the Integration. Refer to the CATIA documentation for CAA (compiler version, Visual Studio version …) prerequisites.

Compilation of **CATIA 32-bit** can be performed only on Windows 32-bit. Compilation of **CATIA 64-bit** can be performed only on Windows 64-bit.

CATIA 32-bit on Windows 64-bit only works in the following environment set-up:

- Build the application on a 32-bit Windows platform.
- Copy the workspace to the 64-bit Windows platform.
- Set the _MkmkOS_BitMode variable to 32 (`set_MkmkOS_BitMode=32`).
- Launch CATIA 32-bit.

2.4 Shared Library Clean-up

When Teamcenter stops using a module/library on AIX, the module/library is not *automatically* removed from the shared library region. In some cases, this prevents Teamcenter Integration for CATIA V5 from overwriting libraries.

Libraries, by default, remain in memory unless the system is rebooted or the **slibclean** command is performed.

It is recommended to run the slibclean command prior to installing the Integration. It is **not** recommended to run this command using InstallAnywhere.

The slibclean command must be performed by the *superuser*.

2.5 Localization / Internationalization

4-Tier Teamcenter Integration for CATIA V5 installation can be deployed with a Client and Server running in different locales.

For example, Teamcenter Integration for CATIA V5 Client can be installed on a French operating system while Teamcenter Integration for CATIA V5 Server is located in an English environment.

3.0 Getting Started

Teamcenter Integration for CATIA V5 integrates Siemens PLM Software's **Teamcenter** and **Active Workspace** with Dassaults Systemes' **CATIA V5** software.

Teamcenter and CATIA V5

The Integration combines the capabilities of Teamcenter and CATIA V5 products to enable users to place CATIA components (products and parts) and drawings under Teamcenter control. Use the Integration to:

- Create and edit CATIA V5 models,drawings and assembly relationships
- Perform partial load and load merge for efficient assembly management
- Create new product revisions or version updates for work in progress
- Generate drawing title blocks
- Manage mechanical properties of CATIA models and assemblies
- Store, load and migrate CATIA V4 data to CATIA V5.

Teamcenter powers innovation and improves productivity by connecting people with the product and process knowledge they need to effectively function in a globally oriented product lifecycle. Teamcenter’s proven digital lifecycle management solutions are built on an open PLM foundation. Teamcenter is the world’s most widely used PLM system. It is backed by Siemens PLM Software’s leadership in delivering Global Innovation Networks that enable companies to make unified, information-driven decisions at every stage in the product lifecycle.

Teamcenter manages engineering data and processes in a secure, distributed environment. All engineering data, such as models, part structures and technical specifications are stored in a relational database. The user interface is menu driven and provides comprehensive management capabilities.

Active Workspace and CATIA V5

The Integration combines the capabilities of **Active Workspace** and CATIA V5 products to enable users to place CATIA components (products and parts) under **Embedded** Active Workspace control. Use the Integration to:

- Load an assembly selected in the Active Workspace in CATIA V5
- Load a document selected in CATIA into Active Workspace
- Highlight an instance selected in CATIA in Active Workspace
- Copy an instance in Active Workspace and paste it into an assembly in CATIA
- Copy an instance in CATIA and paste it into an assembly in Active Workspace
- Check In/Check out an item revision

- Create CATPart/CATProduct datasets
- Cross Refresh.

Under **Standalone** Active Workspace, use the Integration to:

- Load an assembly selected in the Active Workspace in CATIA V5
- Load a document selected in CATIA into Active Workspace.

3.1 Prerequisites for System and Application Administrators

The System Administrator must ensure the minimum system requirements have been met.

The Application Administrator must perform the following steps before using Teamcenter Integration for CATIA V5.

1. Set up the Database Schema.
2. Establish Teamcenter user accounts.
3. Customize the Title Block.

3.2 Installation Overview

Best Practice

Perform the following steps before proceeding with the installation to help ensure a successful installation process.

1. Uninstall previous versions of Teamcenter Integration for CATIA V5.
2. Purge Teamcenter's RAC.

A clean installation of Teamcenter Integration for CATIA V5 results from the following steps.

1. Meet the prerequisites.
2. Customize Teamcenter files for Windows or UNIX.
3. Create the Teamcenter Integration for CATIA V5 directories for Windows.

4.0 The Installation Process

4.1 Installation Summary

The installation process performs the steps listed below when applying the template.

The Teamcenter Integration for CATIA V5 environment is **case-sensitive**. Refer to the Integration's Database Schema documentation, if necessary.

4.1.1 Database Configuration

1. The **CatiaExport** tool is created.
2. **Datasets** are created:
    - CATPart
    - CATProduct
    - CATSecondaryProduct
    - CATSecondaryPart
    - CATDrawing
    - catia
    - catiaSecondary
    - DirectModel
    - CATCgm
    - CATWrl
    - CATAnalysis
    - CATAnalysisComputations
    - CATAnalysisResults
    - CATCache
    - CATProcess
    - CATShape

3. A **BOM View** type named catia is created.
4. **CATIA Notes** types are created:
   - catiaOccurrenceName
   - catiaParentPartName
   - catiaFileName
   - catiaOccurenceOEM
   - catiaOccurenceDesc
5. **CATIA Relations** types are created:
   - catiaV5_MML
   - catiaV5_DWGLink
   - catiaV5_MMLAssy
   - catia_MFGLink
   - catia_alternateShapeRep
   - catia_analysisFilesLink
   - catia_analysisLink
   - catia_auxiliaryLink
   - catia_cache_link
   - catia_componentShapeRep
   - catia_shapeProperties_link
   - catia_secondary_link
6. **Forms** are created:
   - catia_model_attributes
   - catia_assembly_attributes

- catia_doc_attributes
- catia_published_element
- catia_shapeProperties
- catImport_attributes

7. **POM Classes** are created:
    - POM_catImport_attr_form
    - POM_catiaAssyAttributes
    - POM_catiaModelAttributes
    - POM_catia_absocc
    - POM_catia_analysis_userdata
    - POM_catia_doc_attr_form
    - POM_catia_dwg_userdata
    - POM_catia_mfg_userdata
    - POM_catia_mmlAssy_userdata
    - POM_catia_mml_userdata
    - POM_catia_publication_form
    - POM_catia_shapeProperties
    - POM_componentSR_userdata
8. Items to contain the datasets for seed documents are created.
    - IMCATIAPart
    - IMCATIAProduc
    - IMCATIADrawing

4.1.2 CATIA Configuration

1. The **Teamcenter Integration for CATIA V5** menu is created.

4.2 InstallAnywhere Features

Use **InstallAnywhere** to guide the Integration installation process.

Java Virtual Machine (JVM)
Localization (language preference)
Installer Modes (GUI or Silent)
Response (Properties) Files (text files providing installer settings)

4.2.1 Java Virtual Machine (JVM)

Java Virtual Machine (JVM) is installed on the client by InstallAnywhere and is required for successful Integration installation. The JVM is also required during the uninstallation process.

Removing the JVM, installed by default, causes the client uninstallation process to fail.

4.2.2 Localization

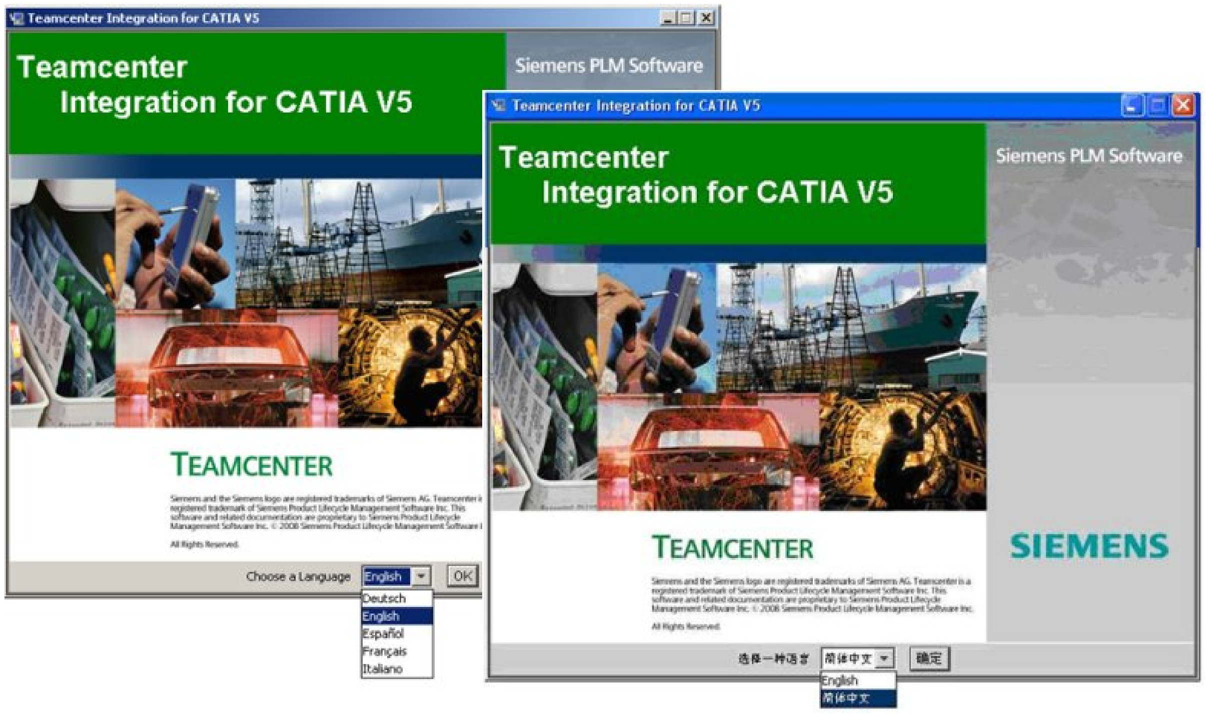
InstallAnywhere is localized permitting dialogs and panels to display in one of ten languages; English, French, Italian, Spanish, German, Russian, Korean, Simplified and Traditional Chinese, Japanese.

The available languages include the default system language detected on the target machine and English.

Refer to the following table for available language choices.

System Language	InstallAnywhere Installation Translations

System Language	InstallAnywhere Installation Translations
English, French, Italian, Spanish, German	English, French, Italian, Spanish, German
Russian	Russian, English
Korean	Korean, English
Japanese	Japanese, English
Simplified Chinese	Simplified Chinese, English
Traditional Chinese	Traditional Chinese, English



Siemens and the Siemens logo are registered trademarks of Siemens AG. Teamcenter is a registered trademark of Siemens Product Lifecycle Management Software Inc. This software and related documentation are proprietary to Siemens Product Lifecycle Management Software Inc. © 2008 Siemens Product Lifecycle Management Software Inc.

All Rights Reserved.

4.2.3 Installer Modes

There are two methods of installation:

1. **Graphical User Interface (GUI)**. This method uses Wizard panels and dialog boxes and requires user interaction.

2. **Silent**. This method requires no user interaction. All settings are provided in a Response (Properties) File containing the values required by InstallAnywhere to control the installation. The silent mode is fully supported on all platforms.

Response files are text files containing settings for an installer (*installer properties file*) and are generated by capturing user responses when executing an installer. A response file must be created prior to using the silent mode installation method.

The response file must not contain variables, such as `$ HOME`, `% temp`.

Perform the following steps to initiate the silent installer enabling InstallAnywhere to run without user interaction, i.e, in **Silent Mode**.

1. Create a response file by entering the following in a command line: `<Installer name> -r <full path to the response file destination>`
   The response file is created at the end of the installation at the given location.

   For example, `myinstaller.exe -r c:\temp\myresponse.properties` writes the response file, `myresponse.properties`, to the temp directory on c:\.

2. Launch InstallAnywhere in silent mode by entering the following in a command line: `<Installer name> -i silent -f <full path of the response file location>`

   A *direct* or *relative path* can be used for the response file, for example, `myinstaller.exe -i silent -f myresponse.properties`

4.3 Over the Web Installation (OTW)

Different types of clients can be installed using Teamcenter Integration for CATIA V5 Over the Web (OTW) installation:

- RAC client installation (default)
- Teamcenter Embedded Active Workspace installation
- Teamcenter Integration for CATIA Loader installation
- Teamcenter Standalone Active Workspace installation

A Teamcenter Client Communication System is required when a CATIA Loader or Active Workspace installation is deployed without a Rich Client installation.

Deployment requirements depend on whether it is a 4-Tier Portal, Teamcenter Integration for CATIA Loader deployment or an Active Workspace deployment.

OTW installation involves four steps:

1. Create a Distribution Server
2. Create a Distribution Server Instance
3. Install Teamcenter Integration for CATIA V5
4. Client Deployment

Using OTW installation, create a *%WORK_DIR%* directory for **both** the current user that connects to the Integration and the administrator performing the installation or set a value such as *%%USERNAME%%* (using double %) instead of *%USERNAME%* in INSWEB for WORK_DIR to create **different** directories.

4.3.1 Overview

Use the OTW installation method to create a Teamcenter installation that will be deployed the same way on every client machine.

- Administrators fully pre-configure the installation.

- Client deployment is required to launch the Teamcenter 4-Tier Portal installer, including the Integration, the Teamcenter Integration for CATIA Loader installer or the Active Workspace installer.

Refer to the **Administrator** section of this document for information on administrator configuration.

4.3.2 Prerequisites

Strong familiarity with **Insweb** (Teamcenter Web Application Manager) and **Teamcenter Client Communication System** are required for successful installation. Refer to Teamcenter Help documentation for information on Teamcenter Web Application Manager, if necessary.

4.3.3 Administrator Configuration

Installation on 4-tier Teamcenter clients use **Insweb** (Teamcenter Web Application Manager). Insweb should be installed on a server-side machine with administrator privileges to enable the administrator to install a Rich Client Distribution Server, build Rich Client Distributions and install the 4-tier Rich Client over a network onto Client Hosts.

4.4 Installing Translation Services

It is strongly recommended that installation of Translation Solutions be performed using a dba user. Failure to do so prevents preferences from being imported into Teamcenter.

There are three translation scripts based on Teamcenter's *Dispatcher* translation management capabilities.

1. **SIEMENS.catiav5tojtdirect**; used to convert CATIAV5 files into JTs.

2. **SIEMENS.catiav4tojtdirect**; used to convert CATIAV4 files (.model) into JTs.

3. **SIEMENS.catiav5preview**; used to create thumbnails (.bmp files) from CATIA files *(Windows platform only)*.

Two *Dispatcher* components, DispatcherClient and Module (part of the Dispatcher Server), must be modified to host and run the translators. These components may be installed on different machines.

Refer to Teamcenter documentation for information on using *Dispatcher*.

Translators used by the SIEMENS.catiav5tojtdirect, SIEMENS.catiav4tojtdirect and SIEMENS.catiav5preview scripts are Java based client applications using Teamcenter Service-Oriented Architecture (SOA). Refer to Teamcenter Services' documentation, if necessary.

Use InstallAnywhere to deploy the required translator files, import related preferences and edit customization files and scripts.

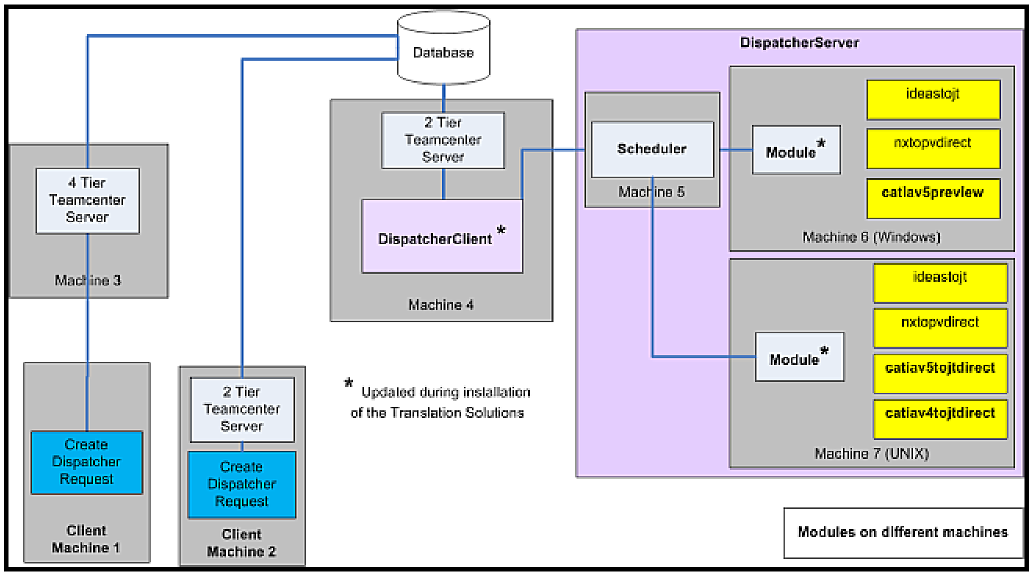
4.4.1 Deployment Strategies

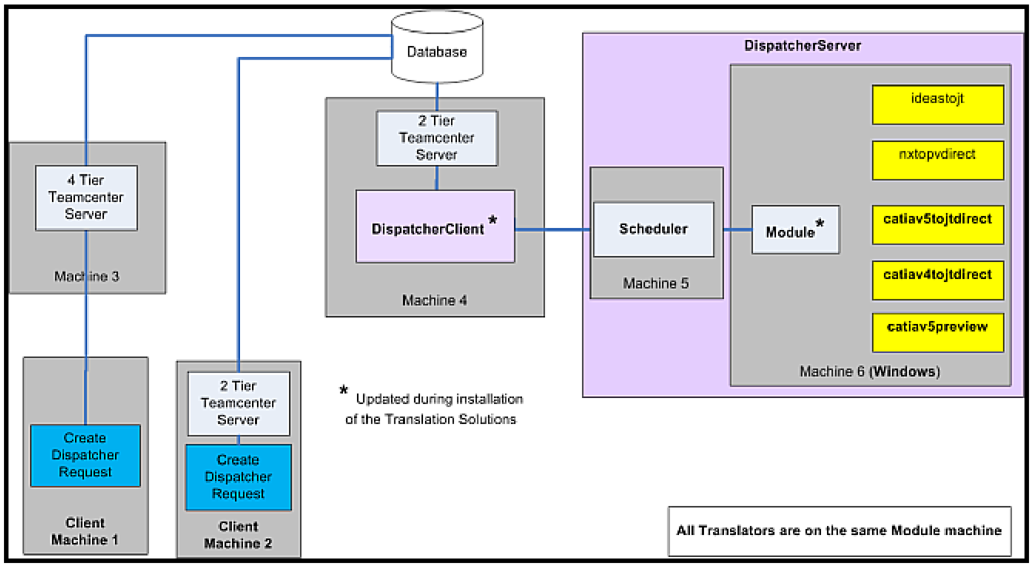
Installation of **SIEMENS.catiav5tojtdirect**, **SIEMENS.catiav4tojtdirect** and **SIEMENS.catiav5preview** deploys files on the DispatcherClient and the Module machine of the Dispatcher Server.

SIEMENS.catiav5preview is only available for Windows platforms, therefore, it may be necessary to redefine the way Dispatcher components are distributed.

A Dispatcher Server distributed on separate hosts may result in the deployments illustrated below.

In both cases, the DispatcherClient and Module(s) are updated during the Translation Solutions installation process.





4.4.2 Installation Prerequisites

To convert CATIA files into JT files

To create .bmp thumbnails from CATIA Files

It is strongly recommended that installation of Translation Solutions be performed using a dba user. Failure to do so prevents preferences from being imported into Teamcenter.

Ensure the following prerequisites have been met prior to installing Translation Services.

1. Familiarity with Teamcenter Service-Orientated Architecture (SOA) and Teamcenter Dispatcher.

2. **FMS**, with FMS client access, available on the Module machine.

3. Stop all Dispatcher services prior to installing Translation Solutions.

4. Define a user dedicated to translations. This user must have sufficient privileges to be able to export CAD data files under Teamcenter datasets and import the translated files into Teamcenter.

5. Meet the specific installation prerequisites required to convert CATIA files into JT files and/or create thumbnails (.bmp files) from CATIA files.

4.4.2.1 To convert CATIA files into JT files (SIEMENS.catiav5tojtdirect and SIEMENS.catiav4tojtdirect)

In addition to installation prerequisites above, ensure a JT translator utility is installed on the Module machine.

4.4.2.2 To create .bmp Thumbnails from CATIA Files (SIEMENS.catiav5preview)

SIEMENS.catiav5preview is only available on 32 and 34 bit Windows.

The SIEMENS.catiav5preview translator uses a utility, *getmini.exe*, to create thumbnails out of CATIA files. This utility is deployed during the installation process on the Module.

In addition to the installation prerequisites above:

1. CATIA V5 must be installed on the machine hosting the getmini.exe file (Module machine).

2. The **same** version of CATIA V5 used on the client machine of the Integration must be the same version used on the machine hosting the getmini.exe.

3. On the Integration client machines, ensure the **Do not activate default shapes on open** option located in *Options\Infrastructure\Product Structure\Product Visualization* is not toggled on. Toggle this option to prevent thumbnails from being created during the translation process resulting in a blank .bmp.

4.4.3 Using InstallAnywhere

Perform the following steps to install Translation Solutions using InstallAnywhere.

Review the InstallAnywhere Features section before proceeding.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

1. Double-click on the InstallAnywhere icon or use a command line to launch InstallAnywhere, for example on Windows : `C\InstallerName.exe.`

2. Choose the Installation language, then pick **OK**.

3. Review the installation procedure message in the InstallAnywhere Introduction window to ensure you are ready to proceed with the installation, then pick **Next** to continue.

4. Pick **Translation Solutions,** then pick **Next**. Refer to the InstallAnywhere documentation for UNIX or Windows for information on installing other features.

5. Pick a translator(s) to enable, then pick **Next**.

6. Choose the Dispatcher component(s) to update *on the current machine*, then pick **Next**.

The Dispatcher Server can not be updated to enable the thumbnails translation service on UNIX machines.

7. Enter the Dispatcher Root directory path, then pick **Next**. The path must contain the following directory:

   - **DispatcherClient** to update the DispatcherClient
   - **Module** to update the Dispatcher Server.
   - **DispatcherClient** *and* **Module** to update both DispatcherClient and Server.

Two separate installations are required to update two Dispatcher components when the DispatcherClient and Module exist in different locations on the same machine.

8. Preferences related to Translation Services are imported during the installation process using Teamcenter's preferences_manager utility. The import occurs during the *MERGE* mode; the scope is set to *SITE*.
   Enter the user ID and password of the user with administrative privileges (usually *infodba*) then pick **Next** to continue.

9. Teamcenter Service-Oriented Architecture (SOA) Java applications deployed on the Module component of the Dispatcher Server must have access to an existing FMS installation location. Refer to Teamcenter Services documentation, if necessary.

> FMS .jar libraries are shipped with FMS installations and are imported into the Integration's code.

10. Enter the path to the FMS installation located on your machine, then pick **Next** to continue. This field will be automatically populated with the value in the *FMS_HOME* environment variable, if available. The value entered must contain the .jar subdirectory which contains the following files:

    a. fccjavaclientproxy.jar

    b. fmsclientcache.jar

    c. fmsutil.jar

    d. fscjavaclientproxy.jar

11. The CATIA Architecture panel displays when the system architecture is 64 bit (Windows 64 bit). Pick the CATIA architecture required, then pick **Next** to continue.

12. Review the Pre-Installation Summary, Installing and Install Complete screens to ensure the information detailed in these screens matches your current installation.

4.4.4 Post-Installation Steps

Set the Teamcenter Service Host Address and Dedicated User (Dispatcher Server)
Modify the .catiaV5tojtdirect Script (Dispatcher Server)
Modify the .catiaV4tojtdirect Script (Dispatcher Server)

4.4.4.1 Set the Teamcenter Service Host Address and Dedicated User (Dispatcher Server)

Configure client applications to use Teamcenter Services by indicating how to connect to a running instance of Teamcenter server. Perform the following steps as Administrator:

1. Locate the following config.properties files on the Module component of the Dispatcher Server:

a. Module\Translators\catiav5tojtdirect\config.properties

b. Module\Translators\catiav4tojtdirect\config.properties

c. Module\Translators\catiav5preview\config.properties

2. Enter the following information in the **config.properties** file for each translator (catiav5tojtdirect, catiav4tojtdirect and catiav5preview) installed:

   a. **DB_Host**; specify the Teamcenter server address

   b. **user.username**; the user dedicated to translations

   c. **user.password**; the password for the user dedicated to translations

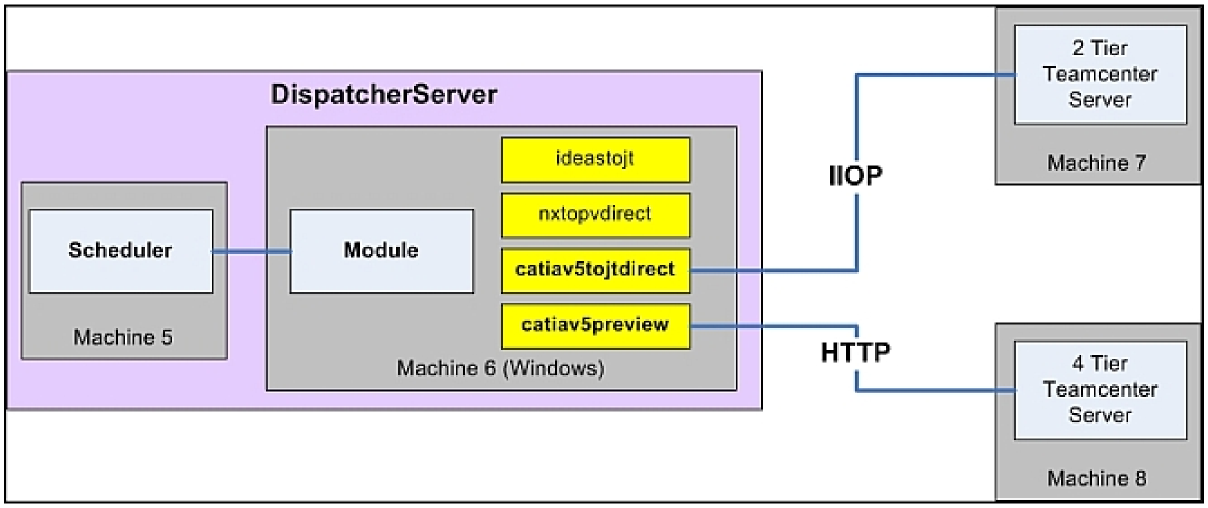
   d. **user.group**; optional attribute related to the user dedicated to translations. Default values are used if unedited.

   e. **user.role**; optional attribute related to the user dedicated to translations. Default values are used if unedited.

About DB_Host

SOA client applications support both 2-Tier (IIOP) and 4-Tier (HTTP) deployments.

In the example below, the JT translator application (catiav5tojtdirect) connects to a 2-Tier Teamcenter server and the thumbnail translator application (catiav5preview) connects to a 4-Tier server.



The **connection type** (HTTP or IIOP) is set by the DB_Host entry in the config.properties file:

- 4-Tier Teamcenter server connections: *DB_Host=http://server:port/app-name*

- 2-Tier Teamcenter server connection: *DB_Host=iiop:server:port/server-id*

For **IIOP**: a common DB_Host string is *iiop:localhost:1572/TcServer1*. In this case, use TEM to set the iiop configuration for 2-Tier Client to **PER_CLIENT**.

Refer to the following definitions and examples as described in the Teamcenter Service Guide Documentation

Term	Definition
server	Network name of the server machine. For 4-Tier deployments, this is the host machine the Teamcenter Web tier is executing on. For 2-Tier deployments, this is the host machine the Teamcenter server is executing on.
port	Port number on the Teamcenter machine that is configured for communications. For HTTP communications, this is the port number on the Web application server. For example, if you are using WebLogic, the default value is 7001, or if you are using Apache Tomcat, the default value is 8080. For IIOP communications, this is the port defined by the CORBA ORB Endpoint.
server-id	CORBA ID of the Teamcenter server
app-name	Application name that the Teamcenter Web tier processes are deployed under. The default value is tc but may have been changed during the installation and configuration of the Teamcenter server software.
service-port	Name of the Teamcenter Services operation being called. The optional URL query string wsdl returns the service WSDL, while the URL without the query string executes the service request

Examples

- iiop:192.168.2.54:1154/TcServer1
- iiop:localhost/TCS2
- http://192.168.25.19:8080/tc

4.4.4.2 Modify the .catiav5tojtdirect Script (Dispatcher Server)

As Administrator, modify the **convert_catiav5tojt** script located in *Module\translators\catiav5tojtdirect* based on your specific requirements relating to the JT utility.

The convert_catiav5tojtdirect script shipped with the Translator is an example which runs the default JT Translator utility, Theorem.

4.4.4.3 Modify the .catiav4tojtdirect Script (Dispatcher Server)

As Administrator, modify the **convert_catiav4tojt** script located in *Module\translators\catiav4tojtdirect* based on your specific requirements relating to the JT utility.

The convert_catiav4tojtdirect script shipped with the Translator is an example which runs the default JT Translator utility, Theorem.

4.4.5 Verifications

.catiav5preview
SIEMENS.catiav4tojtdirect
SIEMENS.catiav5tojtdirect
Preferences

Error and warning messages display during the installation process if problems occur. Review the log files in the final panel of the installer for additional information.

4.4.5.1 Verifications: .catiav5preview

Error and warning messages display during the installation process if problems occur. Review the log files in the final panel of the installer for additional information.

4.4.5.1.1 DispatcherClient

Ensure the following has been deployed on the DispatcherClient machine:

1. The Cat2DCPreview.jar file has been copied to the *DispatcherClient\lib* directory.
2. Cat2Preview has been added to the import line of *DispatcherClient\conf\Service.properties*
3. The cat2preview_env.xml file has been copied to *DispatcherClient\install* and the preferences listed in this file have been successfully imported in Teamcenter by the Teamcenter preferences_manager utility. Refer to the Preferences section.

4.4.5.1.2 Module

Ensure the following has been deployed on the Module machine (Windows only):

1. Ensure *Module\Translators\catiav5preview* has been created and it contains:
   a. *Cat2DSPreview.jar*
   b. *catiav5preview.bat*
   c. *getmini.exe* (the thumbnail generator utility)
   d. config.properties
2. Ensure *Module\conf\translator.xml* has been edited and .catiav5preview has been added.
3. Ensure *Module\Translators\cat2common\soa_lib* has been created and it contains Teamcenter SOA client libraries.

Refer to the Troubleshooting section for additional help.

4.4.5.2 Verifications: SIEMENS.catiav4tojtdirect

Error and warning messages display during the installation process if problems occur. Review the log files in the final panel of the installer for additional information.

4.4.5.2.1 DispatcherClient

Ensure the following has been deployed on the DispatcherClient machine:

1. The Cat2DCTransJTV4.jar has been copied to the *DispatcherClient\lib* directory.
2. Cat2DCTransJTV4 has been added to the import line of *DispatcherClient\conf\Service.properties*
3. The cat2transjtv4_env.xml has been copied to *DispatcherClient\install* and the preferences listed in this file have been successfully imported in Teamcenter by the Teamcenter preferences_manager utility. Refer to the Preferences section.

4.4.5.2.2 Module

Ensure the following has been deployed on the Module machine:

1. Ensure *Module\Translators\catiav4tojtdirect* has been created and it contains:
   a. *Cat2DSTransJTV4.jar*
   b. *catiav4tojtdirect.bat* (or .sh)
   c. *convert_catiav4tojt.bat* (or .sh)
   d. config.properties
2. Ensure *Module\conf\translator.xml* has been edited and .catiav4tojtdirect has been added.
3. Ensure *Module\Translators\cat2common\soa_lib* has been created and contains Teamcenter SOA client libraries.

4.4.5.3 Verifications: SIEMENS.catiav5tojtdirect

Error and warning messages display during the installation process if problems occur. Review the log files in the final panel of the installer for additional information.

4.4.5.3.1 DispatcherClient

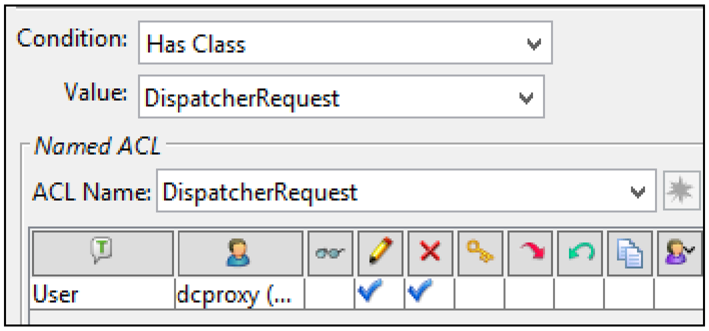
Ensure the following has been deployed on the DispatcherClient machine:

1. The Cat2DCTransJT.jar has been copied to the *DispatcherClient\lib* directory.

2. Cat2DCTransJT has been added to the import line of *DispatcherClient\conf\Service.properties*

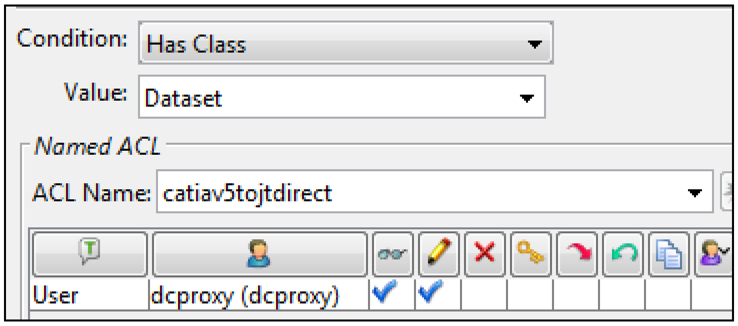
3. The cat2transjt_env.xml has been copied to *DispatcherClient\install* and the preferences listed in this file have been successfully imported in Teamcenter by the Teamcenter preferences_manager utility. Refer to the Preferences section.

The following ACL must be defined to permit the dcproxy user to process the dataset (DispatcherClient level).


Condition: Has Class
Value: DispatcherRequest
Named ACL
ACL Name: DispatcherRequest

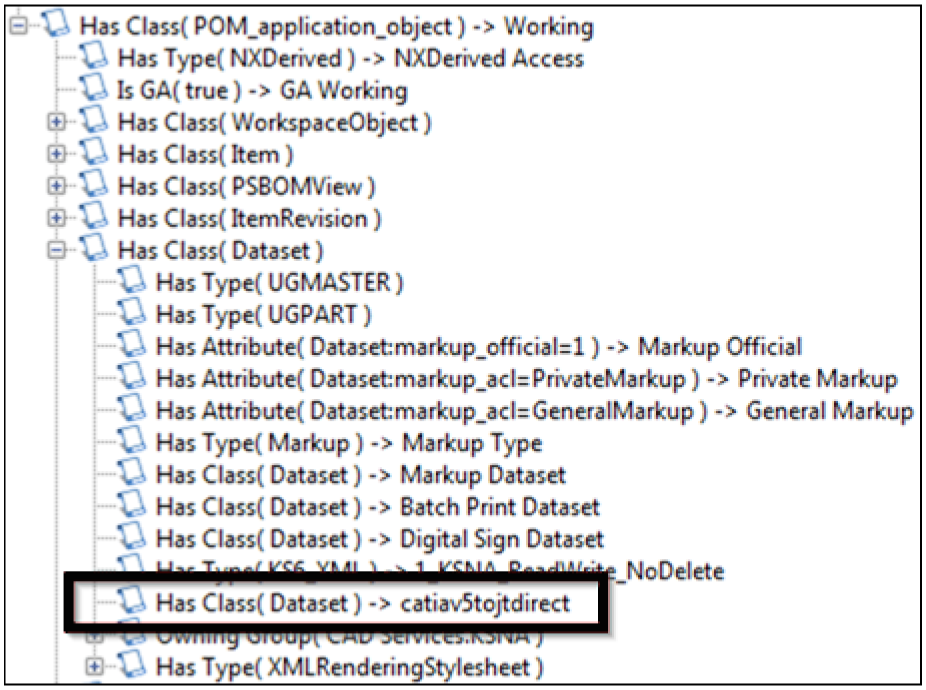
User	dcproxy (...		✓	✓						

Define another ACL so that Named References can be read from the volume sub-folder for non-DBA users.



Condition: Has Class
Value: Dataset
Named ACL
ACL Name: catiav5tojtdirect

User	dcproxy (dcproxy)	✓	✓							



```
Has Class( POM_application_object ) -> Working
    Has Type( NXDerived ) -> NXDerived Access
    Is GA( true ) -> GA Working
    Has Class( WorkspaceObject )
    Has Class( Item )
    Has Class( PSBOMView )
    Has Class( ItemRevision )
    Has Class( Dataset )
        Has Type( UGMASTER )
        Has Type( UGPART )
        Has Attribute( Dataset:markup_official=1 ) -> Markup Official
        Has Attribute( Dataset:markup_acl=PrivateMarkup ) -> Private Markup
        Has Attribute( Dataset:markup_acl=GeneralMarkup ) -> General Markup
        Has Type( Markup ) -> Markup Type
        Has Class( Dataset ) -> Markup Dataset
        Has Class( Dataset ) -> Batch Print Dataset
        Has Class( Dataset ) -> Digital Sign Dataset
        Has Type( KS6_XML ) -> 1_KSNA_ReadWrite_NoDelete
        Has Class( Dataset ) -> catiav5tojtdirect
        Owning Group( CAD Services.KSNA )
        Has Type( XMLRenderingStylesheet )
```

4.4.5.3.2 Module

Ensure the following has been deployed on the Module machine:

1. Ensure *Module\Translators\catiav5tojtdirect* has been created and it contains:

    a. Cat2DSTransJT.jar

    b. catiav5tojtdirect.bat (or .sh)

    c. convert_catiav5tojt.bat (or .sh)

    d. config.properties

2. Ensure *Module\conf\translator.xml* has been edited and .catiav5tojtdirect has been added.

3. Ensure *Module\Translators\cat2common\soa_lib* has been created and contains Teamcenter SOA client libraries.

4.4.5.4 Verifications: Preferences

Error and warning messages display during the installation process if problems occur. Review the log files in the final panel of the installer for additional information.

Preferences related to Translation Services are imported during the installation process using Teamcenter's **preferences_manager** utility. The import occurs during the MERGE mode; the scope is set to SITE. As a result, existing single value preferences are not updated. For example, if the value of the **CATIA_translation_service_name** preference is *SIEMENS.catiatojt*, it will not be updated to *SIEMENS.catiav5tojtdirect*, which is the value required in order to enable the translation solution.

Ensure the preferences have the correct values to meet your specific requirements.

4.5 Installing Teamcenter Integration for CATIA V5 on UNIX

InstallAnywhere
Customizing Teamcenter Files

Ensure the Installation Overview has been reviewed before installing the Integration.

4.5.1 InstallAnywhere

Server Installation
cdrom Extraction
Translation Solutions

Ensure the ksh script shell is installed on UNIX to avoid error messages during the installation process.

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Perform the following steps to install Teamcenter Integration for CATIA V5 files using InstallAnywhere.

1. Double-click on the InstallAnywhere icon or use a command line to launch InstallAnywhere, for example on Windows, `C:\InstallerName.exe.`

2. Pick the Installation language, then pick **OK**.

3. Review the installation procedure message in the InstallAnywhere Introduction window to ensure you are ready to proceed with the installation, then pick **Next** to continue.

4. Pick the feature to be installed.

One or more features may not be available because of your operating system. Refer to Supported Operating Systems for feature availability.

- Pick **Server** to install Teamcenter Integration for CATIA V5 server, then pick here to continue.

- Pick **cdrom Extraction** to extract the files to a user-defined folder, then click here to continue.

- Pick **Translation Solutions** to install translation capabilities, then click here to review specific installation prerequisites, steps and post-installation instructions.

4.5.1.1 Using InstallAnywhere: Server Installation

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Refer to the following steps after **Server Installation** is selected from the InstallAnywhere *Choose Install Set* screen.

1. Indicate the directory where the Teamcenter Server is installed then pick **Next** to continue.

2. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process.

3. Please wait a moment for the installation process to complete.

4. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

The libraries are copied into `$TC_ROOT/lib`. The windows definitions in `$TC_ROOT/lang/motif.client` contain the java classes for Teamcenter.

4.5.1.2 Using InstallAnywhere: cdrom Extraction

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Refer to the following steps after **cdrom Extraction** is selected from the InstallAnywhere *Choose Install Set* screen.

1. Indicate the directory to extract to, then pick **Next** to continue.

2. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process.

3. Please wait a moment for the installation process to complete.

4. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

4.5.2 Customizing Teamcenter Files

The following action is *optional* because the Teamcenter Integration for CATIA(V5) environment performs them automatically.

This action should be performed only when the installation script has failed or when installing a patch.

It is not necessary to copy the library in $TC_ROOT/lib directory. The IMAN_USER_LIB environment variable is defined to set the library paths: `export $IMAN_USER_LIB = $TCICV5_DIR/Teamcenter/lib`.

4.6 Installing Teamcenter Integration for CATIA V5 on Windows

InstallAnywhere
Over the Web (OTW)
Customizing Teamcenter Files
Creating Cache Directories

Teamcenter Integration for CATIA V5 WIN32 contains Integration libraries compiled for **Teamcenter 32bits**. Teamcenter Integration for CATIA V5 WIN64 contains Integration libraries compiled for **Teamcenter 64bits**.

Ensure the Installation Overview has been reviewed before installing the Integration.

4.6.1 Using InstallAnywhere

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Perform the following steps to install Teamcenter Integration for CATIA V5 files using InstallAnywhere.

1. Double-click on the InstallAnywhere icon or use a command line to launch InstallAnywhere, for example on Windows, `C:\InstallerName.exe.`

2. Pick the Installation language, then pick **OK**.

3. Review the installation procedure message in the InstallAnywhere Introduction window to ensure you are ready to proceed with the installation, then pick **Next** to continue.

4. Pick the feature to be installed.

One or more features may not be available because of your operating system. Refer to Supported Operating Systems for feature availability.

- Pick **Client** to install Teamcenter Integration for CATIA V5 client, then pick here to continue.
- Pick **Server** to install Teamcenter Integration for CATIA V5 server, then pick here to continue.
- Pick **Build CATIA Libraries** to generate CATIA Libraries (CAA), then pick here to continue.
- Pick **cdrom Extraction** to extract the files to a user-defined folder, then click here to continue.
- Pick **Translation Solutions** to install translation capabilities, then click here to review specific installation prerequisites, steps and post-installation instructions.

4.6.1.1 Using InstallAnywhere : Client Installation

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Different types of clients can be installed using Teamcenter Integration for CATIA V5 InstallAnywhere:

- RAC client installation (default)
- Teamcenter Embedded Active Workspace installation
- Teamcenter Integration for CATIA Loader installation
- Teamcenter Standalone Active Workspace installation

When selecting Teamcenter Integration for CATIA Loader, Embedded Active Workspace or Standalone Active Workspace installations, FMS HOME must be defined during the installation when a TCCS_SA installation exists.

The Active Workspace MCAD Integration template must be deployed prior to using TcIC with Active Workspace.

Refer to the following steps after **Client Installation** is selected from the InstallAnywhere *Choose Install Set* screen.

1. Enter the directory to copy Integration Client files, the reference directory used when exporting documents as reference, and the directory used for Integration temporary

files. Reference directory default values are *%WORK_DIR%\reference* for Windows and *$WORK_DIR/reference* for UNIX. Pick **Next** to continue.

2. Define the type of Client installation in the Client Selection panel.

    a. Pick **Teamcenter Rich Client** to install a RAC client, then click here to continue.

    b. Pick **Embedded Active Workspace** to install an Active Workspace client in hosted mode, then click here to continue.

    c. Pick **Teamcenter Integration for CATIA V5 Loader** to install a Loader Client, then click here to continue.

    d. Pick **Standalone Active Workspace** to install an Active Workspace client in non-hosted mode, then click here to continue.

3. Define the default client in the list of clients. When multiple clients are defined, the selected client is used by default when a client is not currently running and the command selected in CATIA requires a client. For example, when Rich Client (default) and CATIA V5 Loader installations are installed, the Rich Client is used in priority within the TcIC processes. Change the order of the clients in the *client_env* file to define the CATIA V5 Loader as the default client.

4. Pick an option to install the documentation, then pick **Next** to continue. To skip this step, pick **Next**.

    a. Install new documentation by entering the .zip file path when the **Install Documentation** option was selected from the previous screen.

    b. Enter the directory where the current documentation is installed when the **Use Existing Documentation** option was selected from the previous screen.

5. If **Teamcenter Rich Client** is selected:

    a. Enter the directory where Teamcenter Rich Client is installed, then pick **Next** to continue.

    b. If TC_PORTAL_ROOT is not defined, enter the FMS installation path located on this machine (FMS_HOME) for a TCCS Standalone installation.

    c. Enter the JRE installation path located on this machine (JRE_HOME)

    d. **For Windows installations:** The CATIA Architecture panel displays when the system architecture is **64 bit (Windows 64 bit)**. Pick the CATIA architecture required, then pick **Next** to continue.

6. If **Embedded Active Workspace** is selected:

    a. Enter the Client URL corresponding to the Active Workspace location (optional).

b. If TC_PORTAL_ROOT is not defined, enter the FMS installation path located on this machine (FMS_HOME) for a TCCS Standalone installation or enter the location of the Teamcenter Rich Client Portal folder (TC_PORTAL_ROOT).

> The ActiveWorkspaceHosting.TcIC.URL Teamcenter preference has priority over this parameter (when defined), so this field may be left empty. The Teamcenter Standalone Active Workspace Client URL field will be read when the TcIC_StandaloneAW_Option parameter is set to **true**.
>
> Enter the TC_PORTAL_ROOT path, not the FMS_HOME path, when using FMS from a 4-tiers or 2-tiers Rich Client.

c. Enter the JRE installation path located on this machine (JRE_HOME).

d. Pick **Next** to continue.

7. If **Teamcenter Integration for CATIA V5 Loader** is selected and TC_PORTAL_ROOT or FMS_HOME has not already been defined:

a. Enter the FMS installation path located on this machine (FMS_HOME) for a TCCS Standalone installation, or enter the location of the Teamcenter Rich Client Portal folder (TC_PORTAL_ROOT).

> Enter the TC_PORTAL_ROOT path, not the FMS_HOME path, when using FMS from a 4-tiers or 2-tiers Rich Client.

b. Enter the path to the *SingleEmbeddedView.jar* file of Teamcenter Vis or PLM Vis to enable JT visualization in the Viewer Configuration panel. The path is added to RACLESS_JAVA_CLASSPATH variable. When the path is empty, the JT viewer is not used in TcIC Loader.

c. Enter the JRE installation path located on this machine (JRE_HOME).

d. Pick **Next** to continue.

8. If **Standalone Active Workspace** is selected:

> ***The AWC_TCIC_OpenSupportedTypes and AWC_TCIC_ShowObjectDatasetTypes preferences must be defined to display the Open***

in CATIA command in the Standalone Active Workspace

a. Enter the Client URL corresponding to the Active Workspace location (optional).

b. If TC_PORTAL_ROOT is not defined, enter the FMS installation path located on this machine (FMS_HOME) for a TCCS Standalone installation or enter the location of the Teamcenter Rich Client Portal folder (TC_PORTAL_ROOT).

When Active Workspace (Embedded or Standalone) installation is selected, the windows registry is updated with the *%TCICV5_DIR%\aw\aw_keys.reg* file.

The ActiveWorkspaceHosting.TcIC.URL Teamcenter preference has priority over this parameter (when defined), so this field may be left empty. The Teamcenter Standalone Active Workspace Client URL field will be read when the TcIC_StandaloneAW_Option parameter is set to **true**.

Enter the TC_PORTAL_ROOT path, not the FMS_HOME path, when using FMS from a 4-tiers or 2-tiers Rich Client.

c. Enter the JRE installation path located on this machine (JRE_HOME).

d. Pick **Next** to continue.

9. Pick the CATIA version, enter the path of the CATIA installation, then pick **Next** to continue.

10. Enter the location of your CATIA Libraries (CAA). If the CATIA libraries (CAA) do not exist, return to the Choose Install Sets panel and pick **Build CATIA Libraries** to build them.

11. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process. Please wait a moment for the installation process to complete.

12. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

4.6.1.2 Using InstallAnywhere : Server Installation

Server Installation
Cat2HostedTcIC.war Deployment

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Refer to the following steps after **Server Installation** is selected from the InstallAnywhere *Choose Install Set* screen.

1. Indicate the type of installation to deploy, then pick **Next** to continue.

   a. Toggle **Teamcenter Integration for CATIA server libraries** on to deploy server libraries.

   b. Toggle **Embedded ActiveWorkspace web application** to deploy the .war file under web Server Application for an Embedded Active Workspace installation.

2. Enter the directory where the Teamcenter Server is installed (TC_ROOT) when Teamcenter Integration for CATIA server libraries was selected or enter the folder where the web application file must be deployed when Embedded ActiveWorkspace web application was selected. Pick **Next** to continue.

3. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process.

4. Please wait a moment for the installation process to complete.

5. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

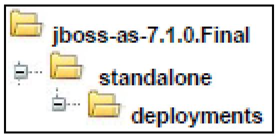
The libraries are copied into `$TC_ROOT/lib`. The windows definitions in `$TC_ROOT/lang/motif.client` contain the java classes for Teamcenter.

4.6.1.2.1 Cat2HostedTcIC.war Deployment

Prerequisite: Installation of the application server.

Jboss

1. **Copy** the *Cat2HostedTcIC.war* file from *%TCICV5_DIR%\Aw* or the local machine where it has been extracted. **Paste** the file into the *jboss-as-7.1.0.Final\standalone\deployments* directory.



2. Restart JBoss and the pool_manager services.

WebLogic

1. Start the WebLogic server administration console.
2. Select **Deployment** from the left pane then pick **Install** from the right pane.
3. In the Install Application Assistant, pick the **Browse** button next to the Deployment Archive box, navigate to the location of the *Cat2HostedTcIC.war* file then pick **Next**.
4. Accept the default **Install this deployment as an application** option and pick **Next**.
5. Pick **Finish** to accept all the default settings, then pick **Save**.
6. Pick **Deployments** then toggle the **Cat2HostedTcIC** application on.
7. Ensure the application state indicates Active and the Health indicates OK.
8. Restart Weblogic and pool_manager services.

Tomcat Application Server

1. Run `Install-location\apache-tomcat-version\bin\startup.bat` to start Tomcat.
2. Pick the Manager App button and log on to the Tomcat Web Application Manager page.
3. Pick **Browse**, then select the *Cat2HostedTcIC.war* file from the File Selection Dialog box.
4. Pick **Deploy**.

WebSphere

1. Start the WebSphere integrated solutions.
2. In the navigation tree, expand Applications and pick **Install New Application.**

3. In the Preparing for the Application Installation pane, enter the path or browse to locate the *Cat2HostedTcIC.war* file.

4. Pick **Open**, then pick **Next**

5. Accept the Fast Path default to install the application, then pick **Next.**

6. Accept the selected installed enterprise application and modules options then pick **Next**.

7. On a single server, non-clustered deployment of the application mapping to the application server already exists, so pick **Next** to continue.

8. The application may need to be mapped to a virtual host depending on how the WAS server is configured. Contact your WAS Administrator, if necessary. In this example, the server uses the default. Pick **Next** to continue.

9. Similar to other WAR file applications, enter a context root name for the application, then pick **Next**.

10. Toggle **metadata-complete attribute**, then pick **Next**.

11. Confirm Application Deployment selections, then pick **Finish**.

12. After the application has deployed, pick **Save** to complete the deployment process.

13. Restart WebSpheree and pool_manager services.

IIS

1. Rename Cat2HostedTcIC.war to **Cat2HostedTcIC.zip**.

2. Create a folder named *Cat2HostedTcIC* inside an existing website published by IIS for AW.

3. **Extract** the contents of the **Cat2HostedTcIC.zip** file into the *Cat2HostedTcIC* folder just created.

4. Restart IIS and pool_manager services.

4.6.1.3 Using InstallAnywhere : Build CATIA Libraries

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

For xPDM (PX1) libraries compilation, set the **TCIC_PX1_COMPILATION** environment variable to **YES** before launching InstallAnywhere. The PX1 - CATIA PPR PDM Gateway Product 1 module must be installed on the client machine where PX1 libraries compilation is launched and on the client machine where the Integration is installed.

Is the PX1-CATIA PPR PDF Gateway Product 1 module installed?	Value of TCIC_PX1_Compilation	Result
no	YES	The *TCIC_PX1_COMPILATION = YES (failed)* error message displays.
yes	YES	The TCIC_PX1_COMPILATION = YES message displays.

Refer to the following steps after Build CATIA Libraries is selected from the InstallAnywhere *Choose Install Set* screen.

1. **For Windows installations**: The CATIA Architecture panel displays when the system architecture is **64 bit (Windows 64 bit)**. Pick the CATIA architecture required, then pick **Next** to continue.
2. Pick the CATIA release used for the Integration. Enter the directory of your CATIA installation, the CATIA RADE directory, then pick **Next** to continue.
3. Indicate the destination folder for built CATIA libraries (CAA).
4. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process.
5. Please wait a moment for the installation process to complete.
6. Review the CATIA libraries build log file then pick **Next** to continue.
7. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

4.6.1.4 Using InstallAnywhere : cdrom Extraction

cdrom Extraction
Merge a Custom Toolbar

Review the InstallAnywhere Features documentation.
Pick Cancel at any time during the process to cancel the installation or pick Previous to return to the previous screen.

Refer to the following steps after **cdrom Extraction** is selected from the InstallAnywhere *Choose Install Set* screen.

1. Indicate the directory to extract to, then pick **Next** to continue.
2. Review the Pre-Installation Summary before continuing, then pick **Install** to initiate the installation process.
3. Please wait a moment for the installation process to complete.
4. Review the installation successful message, then pick **Done** to exit InstallAnywhere. Installation is now complete.

4.6.1.4.1 Merging a Custom Toolbar

The cdrom Extraction process does not extract the full set of binaries so CAA compilation must be performed twice in order to apply a custom toolbar. Merge a custom toolbar by performing the following steps.

1. Compile CAA using InstallAnywhere to get the OOTB compiled CAA.
2. Extract the uncompiled CAA binaries through the cdrom Extraction process.
3. Apply the custom code and compile the toolbar using %TCICV5_DIR%\install\CAACompilation.vbs.
4. Merge CAA.

The steps below use Teamcenter Integration for CATIA V5 (10.0.3). Substitute with your specific information where needed.

Compile CAA using InstallAnywhere

5. Run Teamcenter_Integration_for_CATIAV5_10.0.3_Tc9.1_WIN64.exe.
6. Select **Build CATIA libraries** from the Choose Install Set screen, then pick **Next**.
7. Select the CATIA Architecture then pick **Next**.
8. Enter the directory of the CATIA installation (for example, E:\Program Files\Dassault Systemes\B22) and CATIA RADE paths (for example, E:\Program Files (x86)\Dassault Systemes\T22), then pick **Next**.
9. Enter the Destination folder for CAA. For example, E:\CAA_10.0.3. Pick **Next**, then pick **Install** on the Pre-Installation Summary screen.
10. Review the Install Complete screen, then pick **Done** to complete the CAA complication.

Extract the cdrom using Install Anywhere

11. Run Teamcenter_Integration_for_CATIAV5_10.0.3_Tc9.1_WIN64.exe.
12. Review the Introduction screen then pick **Next** to continue.
13. Select **cdrom Extraction** from the Choose Install Set screen, then pick **Next**.
14. Enter the destination folder used to extract Teamcenter Integration for CATIA V5, then pick **Next** to continue.
15. Review the Pre-Installation Summary screen, then pick **Install** to continue.
16. After extraction has finished, comment out the TLBTcLoadAddin.cpp file from *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R21\CV5CAAToolbar\CV5CAAToolbarTcLoadAddin.m\src*
17. Run compile with CAACompilation.vbs. Enter the release number of CATIA on the CAA Complication box, then pick **OK**.
18. Enter the CATIA installation path, then pick **OK**.
19. Enter the CATIA RADE path then pick **OK** to finish the compilation.

Merge CAA

20. Copy the **CV5CAAToolbar** folder from the *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22* created earlier then paste it in the compiled CAA.

21. Copy/replace all the contents of:

    a. *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22\win_b64\code\bin* into *E:\CAA_10.0.3\CAA\V5R22\win_b64\code\bin*.

    b. *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22\win_b64\control* into *E:\CAA_10.0.3\CAA\V5R22\win_b64\control.*

    c. *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22\win_b64\code\dictionary* into *E:\CAA_10.0.3\CAA\V5R22\win_b64\code\dictionary.*

    d. *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22\win_b64\code\lib* into *E:\CAA_10.0.3\CAA\V5R22\win_b64\code\lib.*

    e. *E:\CDROM_10.0.3\Teamcenter_Integration_for_CATIAV5\CAAObjects\V5R22\win_b64\code\productIC* into *E:\CAA_10.0.3\CAA\V5R22\win_b64\code\productIC.*

22. The merged CAA can now be used for OTW or InstallAnywhere installation using the **Use Existing CAA** option to apply the custom toolbar.

4.6.2 Over the Web (OTW) Installation

The steps detailed in this section are for illustrative purposes only. Please substitute the version number and file names that are applicable to your version of the Integration, as needed.

Review the OTW Overview and Prerequisites before proceeding.

4.6.2.1 Over the Web Installation (OTW): Create a Distribution Server

Refer to Teamcenter Help for information on Distribution Servers. Review the OTW Overview and Prerequisites before proceeding.

The first step in deploying a 4-Tier Teamcenter installation is to create a **Distribution Server**. Perform the following steps to create a Distribution Server.

1. Open the WEB_ROOT directory which contains the Web Application Manager.
2. Launch insweb.
    a. On Windows, double-click the **insweb.bat** file.
    b. On UNIX, use the `insweb` command.
3. Pick **Add...** from the Teamcenter Web Application Manager to add a web application.
4. Complete the Add Application dialog box:
    a. Enter a Name and Staging Location. Pick **Browse...** to locate the correct location, if necessary.
    b. Add the path to the *Web_tier* directory in the Disk Locations for Install Images box, for the Teamcenter software distribution images.
    c. Pick **Distribution Server** from the Solution Type: pull-down list.
5. Pick **Advanced Web Application Options...**.

6. Toggle the **Automatically Build Deployable File** option off, then pick **OK**.

7. Review the Modify Required Context Parameters dialog box.

   a. For most context parameters, accept the default values.

   b. Double-click on the Value box for a given parameter to set the context parameter, then enter a new value.

   c. Select a parameter name to view its description.

8. Pick **OK** on the Modify Required Context Parameters dialog box when finished.

9. View the installation progress in the Progress window, then pick **OK** when completed.

4.6.2.2 Over the Web Installation (OTW): Create a Distribution Server Instance

Perform the following steps to install Distribution Server Instances.

1. Pick **Add...** from the Teamcenter Web Application Manager to add a Distribution Server Instance.

2. Complete the Add Application dialog box:

   a. Enter a Name and Staging Location. Pick **Browse...** to locate the correct location, if necessary.

   b. Enter a disk location for the install images.

   c. Pick **Distribution Server Instance** from the Solution Type: pull-down list. Default Solutions display in the Selected Solutions field.

3. Pick **Advanced Web Application Options...**.

4. Toggle **Automatically Build Deployable File** option off, then pick **OK**.

5. Review the Modify Required Context Parameters dialog box.

   a. For most context parameters, accept the default values.

   b. Double-click on the Value box for a given parameter to set the context parameter, then enter a new value.

   c. Select a parameter name to view its description.

6. Pick **OK** on the Modify Required Context Parameters dialog box when finished.

7. Add a **Host** name in the Modify Required Table - ParentFSCAddressTable dialog box. For example,

   a. Protocol: *http://*

   b. Host: *tcsever* (Teamcenter server host)

   c. Post: *4749* (Teamcenter server port)

   d. Path: may be left blank (optional)

8. View the information displayed in the *Table Description* area, then pick **OK**.

9. Modify parameters on the Modify Required Table - HTTPServerTable dialog box, if necessary. This dialog box displays a list of tier web servers; the first server listed is the default server to login.

10. To view a description of any column , click the column name .then pick **OK**.

11. View the installation progress in the Progress window, then pick **OK** when completed.

4.6.2.3 Over the Web Installation (OTW): Install the Integration Package

Add Teamcenter Integration for CATIA V5 as a Solution for a Distribution Server Instance using the Over The Web (OTW) Installation Package.

The Integration Package contains the following OTW configuration files.

- *teamcenter_integration_for_catiav5.icd*
- .zip files for the operating systems which support 4-Tier Teamcenter: *teamcenter_integration_for_catiav5_ <TcIC_version>_<Tc_version>_otw_ <os_name>.zip*
- *teamcenter_integration_for_catiav5_documentation.icd*
- *Teamcenter_Integration_for_CATIAV5_< TcIC_version>_Documentation.zip*

Perform the following steps to install the Integration Package.

1. Ensure a Distribution Service Instance have been installed.

2. Copy/Paste the *.icd* files into the insweb icd folder.

3. Pick **Reload ICDs** from the Teamcenter Web Application Manager.

4. Pick **OK** on the Progress window after the ICD files have been successfully reloaded.

5. Pick the Distribution Server Instance just created then pick **Modify**.

6. Pick **Modify Disk Locations...** from the Modify Web Application dialog box to add the Integration Package location.

7. Pick **Add...** from the Modify Disk Location dialog box.

8. Enter the Integration Package location in the Disk Location to Add field or pick **Browse...** to locate the correct location, then pick **OK**.

9. Continue adding the necessary files, including available patches, then pick **OK** on the Modify Disk Locations dialog box when finished.

10. Pick **Copy ICD Files...** from the Modify Web Application dialog box.

11. Pick **Yes** to copy the ICD files.

12. View the copy progress in the Progress window, then pick **OK** when completed.

13. Pick **Add Solutions...** from the Modify Web Application Manager to define the solutions to install.

14. Pick **Yes** to confirm and commit the changes before continuing

15. Select **Teamcenter Integration for CATIAV5** to install the Integration and **Teamcenter Integration for CATIAV5 Documentation** to install the online help (optional) from the Add Solutions list, then pick **OK**.

16. Modify the parameter values, if necessary, then pick **OK** when finished. Refer to the Parameter Definitions Table for parameter descriptions and default values.

17. View the installation progress in the Progress window, then pick **OK** when completed.

18. Teamcenter Integration for CATIA V5 has been added as an Installed Solution and has been deployed.

19. Run `start_rmi.bat` and `start_server.bat` to clients access to the installation.

20. The *otw.html* file is the start page for installing 4-Tier Teamcenter Client with the Integration. Provide the *otw.html* file path to clients to launch host installations of 4-Tier Teamcenter Rich Client installations. Because the HTTP Server is already running, install 4-Tiers client on a client workstation by invoking the *otwweb/otw.html* page from a Web browser.

4.6.2.4 Over the Web Installation (OTW): Client Deployment

Summary/Results
Cat2HostedTcIC.war Deployment

Once all configuration steps have been completed by the Administrator, the client must perform the following steps to launch the installer. Refer to the Client Deployment Troubleshooting section, if necessary.

The process must be initiated with root privileges to create the CATIA environment file during the silent installation.

1. Launch the `<otwinstallation_location>/webapp_root/otwweb/otw.html` file
2. Pick **Yes** on the Security warning message to begin the installation process.
3. Wait momentarily until the installation process finishes.
4. Review each informational window to ensure the information is valid, then pick **OK**.

Installation is now complete. Refer to the Summary/Results details below.

Teamcenter 4-Tier Portal, Teamcenter Integration for CATIA Loader or Active Workspace can be used directly after installation; there is no need to restart or logout/login to Windows because an environment variable was not created during the installation.

A Teamcenter Rich Client icon is created on the desktop and a CATIA V5 icon is created when **Create_CATEnv** is the value indicated for the CATIA_Env_Creation parameter.

The *otw.bat* is called in StartRACLessServer.bat to update the installation for Teamcenter Integration for CATIA Loader installations.

4.6.2.4.1 Summary/Results

By default, if no client other than a Rich client is selected, **TcIC_Default_Client** is set to Teamcenter Rich Client. The default client starts when a client is not currently running and a command selected in CATIA requires a client. The value is ignored when it does not correspond to a client defined for the installation. Possible values are Teamcenter Rich Client, Teamcenter Embedded Active Workspace, Teamcenter Integration for CATIA V5 Loader, Teamcenter Standalone Active Workspace.

Rich Client 4-Tier:

- When selected, a RAC client is installed and the Embedded Active Workspace client, Teamcenter Integration for CATIA Loader client or Standalone Active Workspace client can also be installed, if selected.

- When not selected, the Embedded Active Workspace client, Teamcenter Integration for CATIA Loader client or Standalone Active Workspace client can also be installed, if selected.

Teamcenter Embedded Active Workspace

The .war file must be manually deployed under web application server for an Embedded Active Workspace installation.

- When selected, the **TcIC_EmbeddedAW_Option** is set to **true**, the registry must be modified and the Teamcenter Embedded Active Workspace Client URL must be defined. The current user must have sufficient access rights to modify the registry or the registry must link the Teamcenter Embedded Active Workspace Client to Teamcenter Integration for CATIA V5.

The **ActiveWorkspaceHosting.TcIC.URL** Teamcenter preference, when defined, has priority over this parameter, so this field may be left empty. The Teamcenter Embedded Active Workspace Client URL field will be read when **TcIC_EmbeddedAW_Option** is set to **true**.

Teamcenter Integration for CATIA V5 Loader

- When selected, the **TcIC_Loader_Option** is set to **true**.

Teamcenter Standalone Active Workspace

- When selected, the **TcIC_StandaloneAW_Option** is set to **true,** the registry must be modified, and the Teamcenter Embedded Active Workspace Client URL must be defined. The current user must have sufficient access rights to modify the registry or the registry must link the Teamcenter Standalone Active Workspace Client to Teamcenter Integration for CATIA V5.

The **ActiveWorkspaceHosting.TcIC.URL** Teamcenter preference, when defined, has priority over this parameter, so this field may be left empty. The Teamcenter Standalone Active Workspace

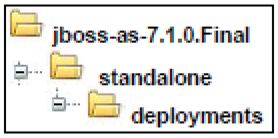
Client URL field will be read when **TcIC_StandaloneAW_Option** is set to **true**.

4.6.2.4.2 Cat2HostedTcIC.war Deployment

Prerequisite: Installation of the application server.

Jboss

1. **Copy** the *Cat2HostedTcIC.war* file from *%TCICV5_DIR%\Aw* or the local machine where it has been extracted. **Paste** the file into the *jboss-as-7.1.0.Final\standalone\deployments* directory.



2. Restart JBoss and the pool_manager services.

WebLogic

1. Start the WebLogic server administration console.

2. Select **Deployment** from the left pane then pick **Install** from the right pane.

3. In the Install Application Assistant, pick the **Browse** button next to the Deployment Archive box, navigate to the location of the *Cat2HostedTcIC.war* file then pick **Next.**

4. Accept the default **Install this deployment as an application** option and pick **Next**.

5. Pick **Finish** to accept all the default settings, then pick **Save**.

6. Pick **Deployments** then toggle the **Cat2HostedTcIC** application on.

7. Ensure the application state indicates Active and the Health indicates OK.

8. Restart Weblogic and pool_manager services.

Tomcat Application Server

1. Run `Install-location\apache-tomcat-version\bin\startup.bat` to start Tomcat.

2. Pick the Manager App button and log on to the Tomcat Web Application Manager page.

3. Pick **Browse**, then select the *Cat2HostedTcIC.war* file from the File Selection Dialog box.

4. Pick **Deploy**.

WebSphere

1. Start the WebSphere integrated solutions.

2. In the navigation tree, expand Applications and pick **Install New Application.**

3. In the Preparing for the Application Installation pane, enter the path or browse to locate the *Cat2HostedTcIC.war* file.

4. Pick **Open**, then pick **Next**

5. Accept the Fast Path default to install the application, then pick **Next**.

6. Accept the selected installed enterprise application and modules options then pick **Next**.

7. On a single server, non-clustered deployment of the application mapping to the application server already exists, so pick **Next** to continue.

8. The application may need to be mapped to a virtual host depending on how the WAS server is configured. Contact your WAS Administrator, if necessary. In this example, the server uses the default. Pick **Next** to continue.

9. Similar to other WAR file applications, enter a context root name for the application, then pick **Next**.

10. Toggle **metadata-complete attribute**, then pick **Next**.

11. Confirm Application Deployment selections, then pick **Finish**.

12. After the application has deployed, pick **Save** to complete the deployment process.

13. Restart WebSpheree and pool_manager services.

IIS

1. Rename Cat2HostedTcIC.war to **Cat2HostedTcIC.zip**.

2. Create a folder named *Cat2HostedTcIC* inside an existing website published by IIS for AW.

3. **Extract** the contents of the **Cat2HostedTcIC.zip** file into the *Cat2HostedTcIC* folder just created.

4. Restart IIS and pool_manager services.

4.6.3 Customizing Teamcenter Files

The following actions are *optional* because the Teamcenter Integration for CATIA V5 environment performs them automatically.

This action should be performed only if the installation script failed or when installing a patch.

Replace some existing Teamcenter files by executing the following instructions.

1. Copy all *lib* files of *%TCICV5_DIR%\Teamcenter\lib in %TC_ROOT%\lib.*

2. *Copy all dll files of %TCICV5_DIR%\Teamcenter\lib in %TC_ROOT%\bin.*

The *%TCICV5_DIR%\env\tcic_var_env.xml* file contains the default value of different variables and can be customized.

4.6.4 Creating Cache Directories

Because some Teamcenter Integration for CATIA V5 functionalities remove all files in a given directory, it is strongly recommended to dedicate the directories listed below to the exclusive use of Teamcenter Integration for CATIA V5.

Ensure there are no spaces within the catuii directory pathname.

1. Create the directories defined in the *%TCICV5_DIR%\install\CATIMAN.vbs* file (*IM_FILES_DIR* and *IM_TMP_DIR*).

2. Create the *%WORK_DIR%\tmp* and *%WORK_DIR%\catuii* directories if default settings are used.

The *%TCICV5_DIR%\install\CATIMAN.vbs* script creates these directories automatically during the install.

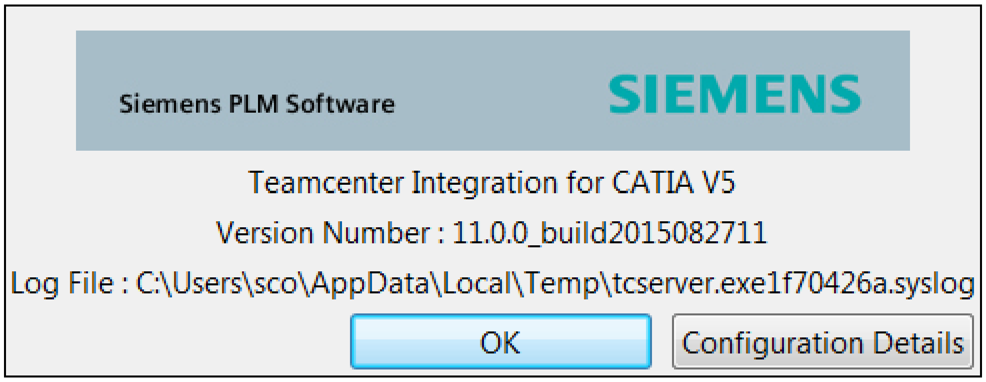
4.7 The About integration Menu within Teamcenter and CATIA V5

In Teamcenter, pick **CATIA V5 >> Help >> About integration**, or in CATIA pick, **Teamcenter >> About integration** to determine the Integration version and other key information related to your installation.

4.7.1 Configuration Details

The **Configuration Details** button on the About Integration panel displays detailed information related to the specific Integration version installed.

In CATIA:

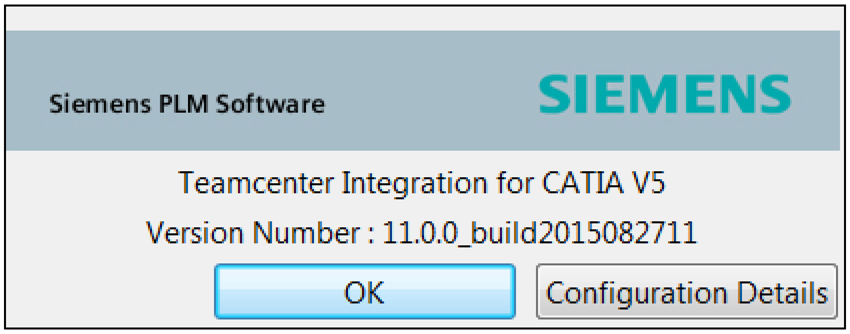

Siemens PLM Software SIEMENS

Teamcenter Integration for CATIA V5

Version Number : 11.0.0_build2015082711

Log File : C:\Users\sco\AppData\Local\Temp\tcserver.exe1f70426a.syslog

In Teamcenter:


Siemens PLM Software SIEMENS

Teamcenter Integration for CATIA V5

Version Number : 11.0.0_build2015082711

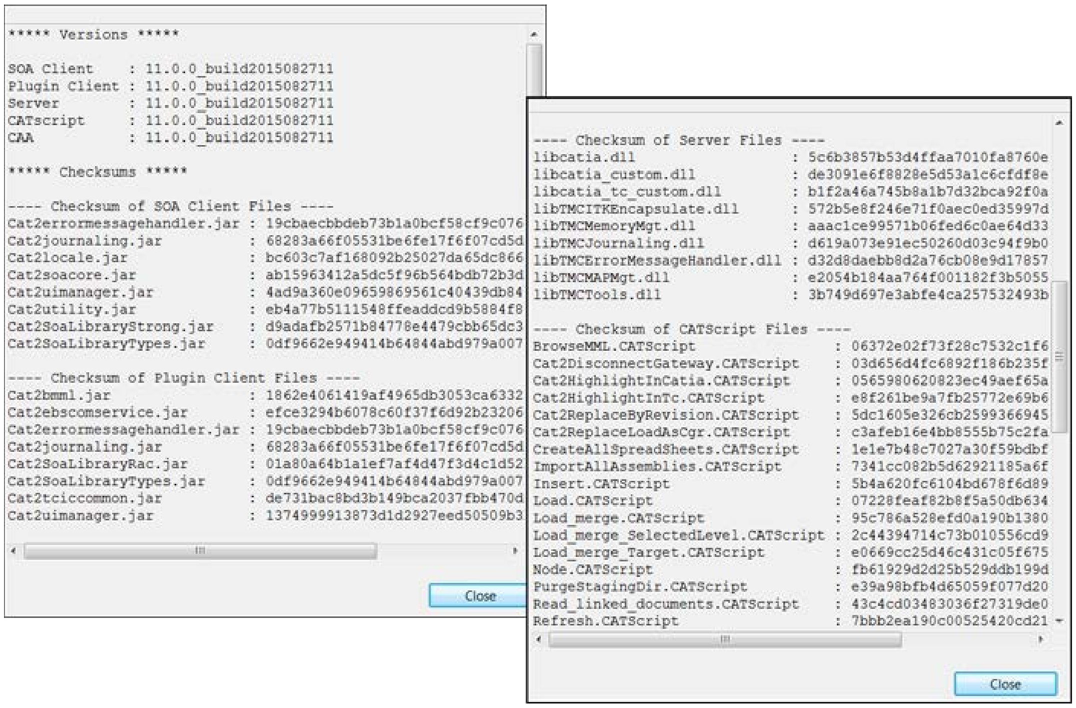
This detailed information may be copied to assist in troubleshooting. The information displayed includes:

- Version Number
    - SOA Client version

    - Plug-in Client version
    - Server version
    - CATScript version
    - CAA version
- Checksums
    - SOA Client files
    - Plug-in Client files
    - Server files
    - CATScript files
    - CATScript batch files

View the checksum information to determine the file modified by a software patch.

The message, *File doesn't exist*, replaces the checksum information when the file does not exist or for Plug-in Client information when Teamcenter RAC client does not launch when the command is launched from CATIA.



```
***** Versions *****

SOA Client    : 11.0.0_build2015082711
Plugin Client : 11.0.0_build2015082711
Server        : 11.0.0_build2015082711
CATscript     : 11.0.0_build2015082711
CAA           : 11.0.0_build2015082711

***** Checksums *****

---- Checksum of SOA Client Files ----
Cat2errormessagehandler.jar : 19cbaecbbdeb73b1a0bcf58cf9c076
Cat2journaling.jar          : 68283a66f05531be6fe17f6f07cd5d
Cat2locale.jar              : bc603c7af168092b25027da65dc866
Cat2soacore.jar             : ab15963412a5dc5f96b564bdb72b3d
Cat2uimanager.jar           : 4ad9a360e09659869561c40439db84
Cat2utility.jar             : eb4a77b5111548ffeaddcd9b5884f8
Cat2SoaLibraryStrong.jar    : d9adafb2571b84778e4479cbb65dc3
Cat2SoaLibraryTypes.jar     : 0df9662e949414b64844abd979a007

---- Checksum of Plugin Client Files ----
Cat2bmml.jar                : 1862e4061419af4965db3053ca6332
Cat2ebscomservice.jar       : efce3294b6078c60f37f6d92b23206
Cat2errormessagehandler.jar : 19cbaecbbdeb73b1a0bcf58cf9c076
Cat2journaling.jar          : 68283a66f05531be6fe17f6f07cd5d
Cat2SoaLibraryRac.jar       : 01a80a64b1a1ef7af4d47f3d4c1d52
Cat2SoaLibraryTypes.jar     : 0df9662e949414b64844abd979a007
Cat2tciccommon.jar          : de731bac8bd3b149bca2037fbb470d
Cat2uimanager.jar           : 1374999913873d1d2927eed50509b3
```

```
---- Checksum of Server Files ----
libcatia.dll                  : 5c6b3857b53d4ffaa7010fa8760e
libcatia_custom.dll           : de3091e6f8828e5d53a1c6cfdf8e
libcatia_tc_custom.dll        : b1f2a46a745b8a1b7d32bca92f0a
libTMCITKEncapsulate.dll      : 572b5e8f246e71f0aec0ed35997d
libTMCMemoryMgt.dll           : aaac1ce99571b06fed6c0ae64d33
libTMCJournaling.dll          : d619a073e91ec50260d03c94f9b0
libTMCErrorMessageHandler.dll : d32d8daebb8d2a76cb08e9d17857
libTMCMAPMgt.dll              : e2054b184aa764f001182f3b5055
libTMCTools.dll               : 3b749d697e3abfe4ca257532493b

---- Checksum of CATScript Files ----
BrowseMML.CATScript                 : 06372e02f73f28c7532c1f6
Cat2DisconnectGateway.CATScript     : 03d656d4fc6892f186b235f
Cat2HighlightInCatia.CATScript      : 0565980620823ec49aef65a
Cat2HighlightInTc.CATScript         : e8f261be9a7fb25772e69b6
Cat2ReplaceByRevision.CATScript     : 5dc1605e326cb2599366945
Cat2ReplaceLoadAsCgr.CATScript      : c3afeb16e4bb8555b75c2fa
CreateAllSpreadSheets.CATScript     : 1e1e7b48c7027a30f59bdbf
ImportAllAssemblies.CATScript       : 7341cc082b5d62921185a6f
Insert.CATScript                    : 5b4a620fc6104bd678f6d89
Load.CATScript                      : 07228feaf82b8f5a50db634
Load_merge.CATScript                : 95c786a528efd0a190b1380
Load_merge_SelectedLevel.CATScript : 2c44394714c73b010556cd9
Load_merge_Target.CATScript         : e0669cc25d46c431c05f675
Node.CATScript                      : fb61929d2d25b529ddb199d
PurgeStagingDir.CATScript           : e39a98bfb4d65059f077d20
Read_linked_documents.CATScript     : 43c4cd03483036f27319de0
Refresh.CATScript                   : 7bbb2ea190c00525420cd21
```

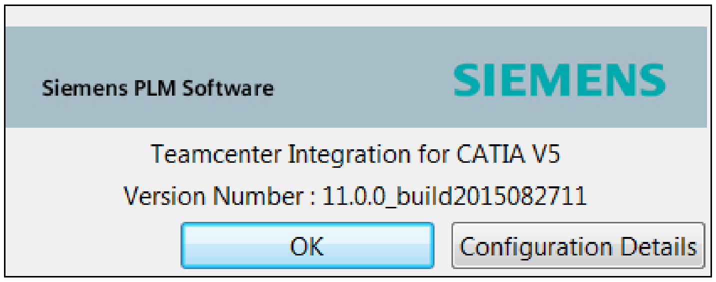
4.7.2 Version Number

There are five different version numbers, based on code, related to your installation. Only the latest installation of the five versions of code displays in About integration.

- SOA Client version
- Plug-in Client version
- Server version
- CATScript version
- CAA version

It is possible to have a different version numbers when a software patch has been installed. For example, when a patch related to the Client version is installed, this version number displays in the About integration panel because it is the most recent version, or latest version (of the five) that has been installed.

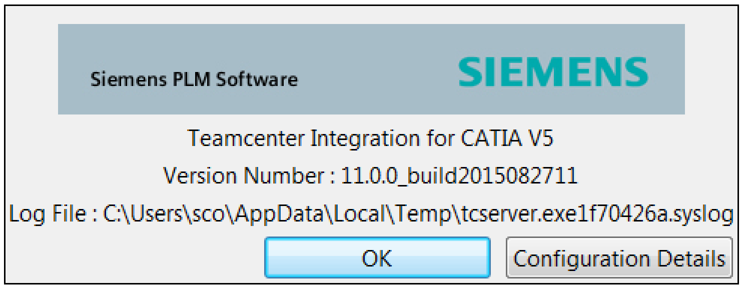
In Teamcenter:



Teamcenter Integration for CATIA V5

Version Number : 11.0.0_build2015082711

In CATIA:



Teamcenter Integration for CATIA V5

Version Number : 11.0.0_build2015082711

Log File : C:\Users\sco\AppData\Local\Temp\tcserver.exe1f70426a.syslog

The *About integration* panel shown indicates the version number is 11.0.0_build2015082711. Pick the Configuration Details button to determine which version/patch was the last to be installed and view other version and configuration details.

5.0 Configuration

5.1 Define Different CATIA Versions to Launch within the Integration

Defining different CATIA versions to use with the Integration requires copying and editing configuration files and shortcuts.

Perform the following steps to permit different versions of CATIA to be launched within the Integration.

The data in the steps assume R18 has been successfully configured using the Installation and R19 is to be added. **Substitute with your specific information where needed**.

1. Copy and Paste the existing Teamcenter_Integration_for_CATIAV5 icon onto the user's desktop.

2. Rename it to a new version, such as *Teamcenter_Integration_for_CATIAV5_R19*.

3. Press MB3 on the renamed icon, then pick **Properties**.

4. View the **Target** (launch command) field within the **Shortcut** tab of the Properties window.

```
"C:\Program Files\Dassault Systemes
\B18\intel_a\code\bin\CATSTART.exe -run "CNEXT.exe" -env
Teamcenter_Integration_for_CATIAV5 -direnv "C:\Documents and
Settings\All Users\Application Data\DassaultSystemes\CATEnv" –
nowindow"
```

5. Modify the `CATSTART.exe` location to the new version.

The location may need to only be changed from *R18* to *R19*, depending on your installation path(s).

6. Copy the *Teamcenter_Integration_for_CATIAV5.txt* environment file located in the directory following `-direnv` to the *Teamcenter_Integration_for_CATIAV5_R19.txt* file.

7. Edit the Target string to reflect the new name.

Assuming the new version of CATIA is installed in the same base path, (only a different Rxx number), open the copied file and replace R18 with R19 and B18 with B19.

8. Save then close the *Teamcenter_Integration_for_CATIAV5_R19.txt* file.

9. Confirm the new Target field displays the new path. If the new version of CATIA is located in an entirely different location, edit the original file accordingly to update the native CATIA path locations. The Integration for CATIA paths will only differ by the Rxx or Bxx.

```
"C:\Program Files\Dassault Systemes
\B19\intel_a\code\bin\CATSTART.exe -run "CNEXT.exe" -env
Teamcenter_Integration_for_CATIAV5_R19 -direnv "C:\Documents and
Settings\All Users\Application Data\DassaultSystemes\CATEnv" -
nowindow"
```

10. Pick **OK** to close the Properties window.

11. Launch CATIA using the new shortcut.

Launching CATIA from Teamcenter defaults to the last version of CATIA installed using the Integration.

5.2 Configure Active Workspace

Define Types of Clients

Embedded Active Workspace Settings and Behavior

Standalone Active Workspace Settings and Behaviors

Load Manager

Open in CATIA

A 4-tiers installation, for Embedded or Standalone Active Workspace, is recommended to better manage structures during Load processes.

The Active Workspace MCAD Integration template must be deployed prior to using TcIC with Active Workspace.

5.2.1 Define Types of Clients

For Embedded Active Workspaces
For Standalone Active Workspaces

The *%TCICV5_DIR%/env/client_env* file defines different clients that can be used. Change the order of clients in this file to display the client to be used in priority.

Embedded Active Workspace is defined as `Client_0` , whereas Standalone Active Workspace is defined as `Client_1`.

The ActiveWorkspaceHosting.TcIC.URL Teamcenter preference, when defined, has priority over the URL defined in the *client_env* file.

```
<client>
Teamcenter Embedded Active Workspace Client
http://vm-tcserver33:7031/awc
Client_0
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
<client>
Teamcenter
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac\otwportal.bat
Client_0
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
<client>
Teamcenter Standalone Active Workspace Client
http://vm-tcserver33:7031/awc
Client_1
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
<client>
Teamcenter Integration for CATIA V5 Loader
C:\TCICV5\racless\TcICLoader.bat
Client_0
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
```

```

```

5.2.1.1 For Embedded Active Workspaces

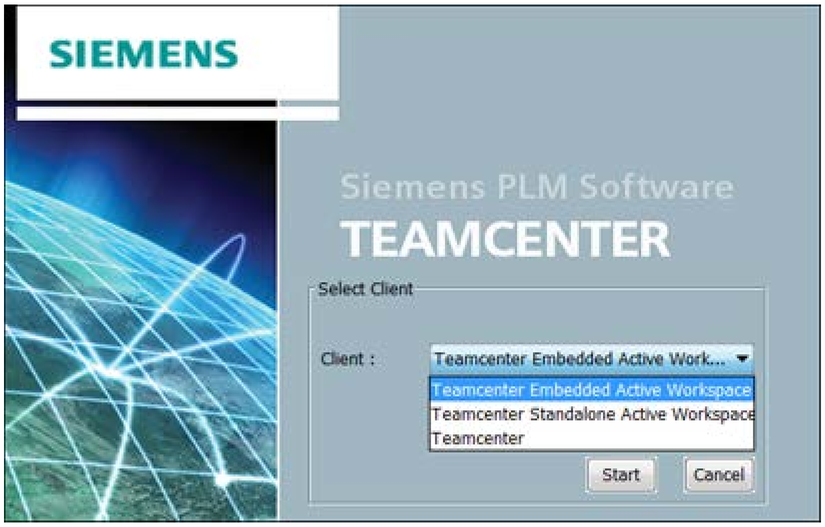
```
<client>
Teamcenter Embedded Active Workspace Client
http://vm-tcserver33:7031/awc
Client_0
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
```

Embedded Active Workspace should always be defined as `Client_0`.

If the Embedded Active Workspace client has been **manually** added to the client_env file, ensure the following:

- Set the TcIC_awhosted_menu variable to **true** in the *%TCICV5_DIR%/env/tcic_var_env.xml* file. The default value is **false**, so Active Workspace menu is not available in the CATIA contextual Menu.
- The Windows registry has to be updated with the *%TCICV5_DIR%\aw\ aw_keys.reg* file.

The Active Workspace home page displays in the CAD web browser inside CATIA. When a RACLess connection has already been established, login information is shared with Active Workspace.


SIEMENS
Siemens PLM Software
TEAMCENTER
Select Client
Client :
Teamcenter Embedded Active Work...
Teamcenter Embedded Active Workspace
Teamcenter Standalone Active Workspace
Teamcenter
Start
Cancel

5.2.1.2 For Standalone Active Workspaces

```
<client>
Teamcenter Standalone Active Workspace Client
```

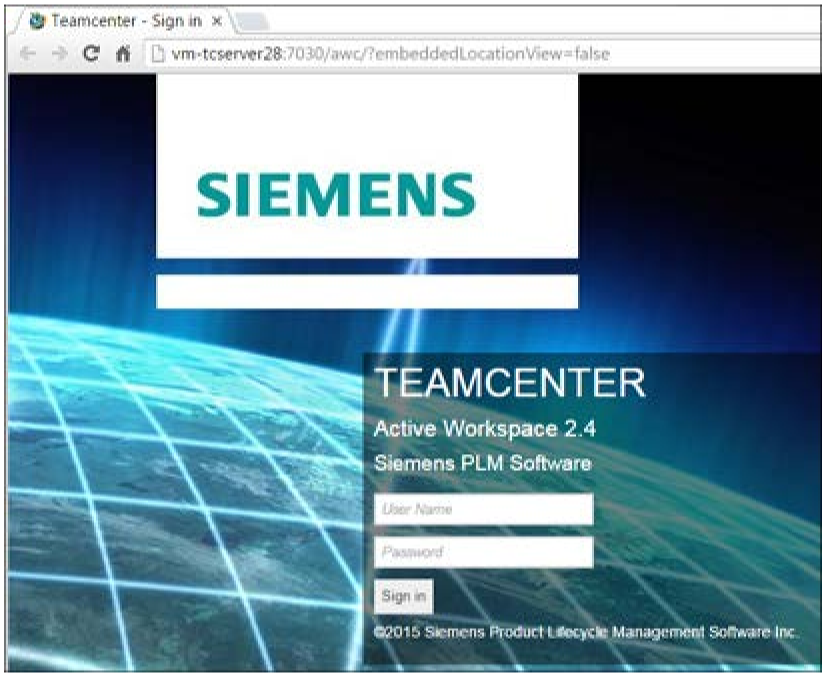
```
http://vm-tcserver33:7031/awc
Client_1
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
```

Standalone Active Workspace should always be defined as `Client_1`.

If the Standalone Active Workspace client has been **manually** added in the *client_env* file, ensure that the Windows registry has been updated with the *%TCICV5_DIR%\aw\ aw_keys.reg* file.

The Active Workspace home page displays in the default web browser outside CATIA.

When the connection is not established, the Active Workspace Login Panel displays to enter User/Group/Role information..


Teamcenter - Sign in
vm-tcserver28:7030/awc/?embeddedLocationView=false
SIEMENS
TEAMCENTER
Active Workspace 2.4
Siemens PLM Software
User Name
Password
Sign in
©2015 Siemens Product Lifecycle Management Software Inc.

5.2.2 Embedded Active Workspace Settings and Location

Default Setting Values

Settings and Locations

Visualization in Active Workspace

Use *%TCICV5_DIR%\env\aw_config.properties* file to configure Active Workspace settings and Embedded Web Browser behaviors.

Host-side native configuration settings in the *aw_config.properties* file are used to configure Active Workspace.

Name	Description
`allowGoHome`	Adds the *jump to home* feature on the User Interface (e.g., gateway). The default value is **true**.
`allowThemeChange`	Permits the User Interface theme to be changed. The default value is **true**.
`allowUserSessionChange`	Permits the user session information to be changed, i.e., users can login/logout as a different users. The default value is **true**.
`hasFullScreenSupport`	Displays the client in full screen mode. The default value is **true** (full screen mode).
`showSiemensLogo`	Displays the Siemens logo. The default value is **true**.
`theme`	The fully qualified theme name requested for use by the host. The default value is "".
`HostSupportsMultipleSelection`	Permits multi-selection in a structure. The default value is **true**.
`PinUnpin`	Defines if the web browser is inside or outside CATIA.

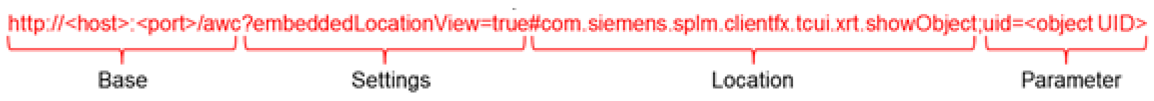
Name	Description
`Highlight`	Displays the Highlight button. The default value is true.
`CheckOut`	Displays the CheckOut button. The default value is true.
`CheckIn`	Displays the CheckIn button. The default value is true.
`CreateCATPart`	Displays the CreateCATPart button. The default value is true.
`CreateCATProduct`	Displays the CreateCATProduct button. The default value is true.
`LoadOptions`	Displays the LoadOptions button. The default value is true.
`LoadCommand`	Displays the LoadCommand button. The default value is true.
`Update`	Displays the Update button. The default value is true.
`ShowInTc`	Displays the ShowInTc button. The default value is true.

5.2.2.1 Default Setting Values

```
Settings=embeddedLocationView=false
Location=com.siemens.splm.clientfx.tcui.xrt.showObject;page=Overvie
w
awhosted.option.AllowGoHome=true
awhosted.option.AllowThemeChange=true
awhosted.option.AllowUserSessionChange=false
awhosted.option.HasFullScreenSupport=false
awhosted.option.ShowSiemensLogo=true
awhosted.option.Theme=com.siemens.splm.clientfx.ui.lightTheme
awhosted.option.HostSupportsMultipleSelection=true
awhosted.PinUnpin.Default=pin
awhosted.PinUnpin.Display=true
awhosted.Highlight.Display=true
awhosted.CheckOut.Display=true
awhosted.CheckIn.Display=true
awhosted.CreateCATPart.Display=true
awhosted.CreateCATProduct.Display=true
awhosted.LoadOptions.Display=true
awhosted.LoadCommand.Display=true

awhosted.Update.Display=true
awhosted.ShowInTc.Display=true
```

5.2.2.2 Settings and Location

The URL uses **Settings** and **Location** values to define the way the item should be opened in Active Workspace.


http://<host>:<port>/awc?embeddedLocationView=true#com.siemens.splm.clientfx.tcui.xrt.showObject;uid=<object UID>
Base
Settings
Location
Parameter

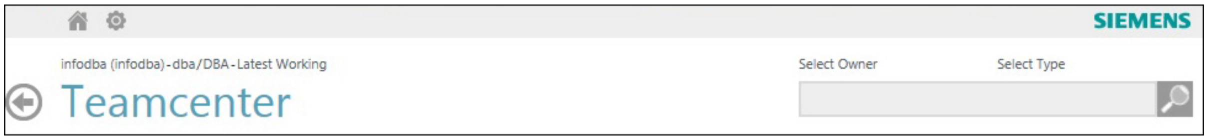
For example, when:

- **Settings**=embeddedLocationView=false
- **Location**=com.siemens.splm.clientfx.tcui.xrt.showObject;page=Overview

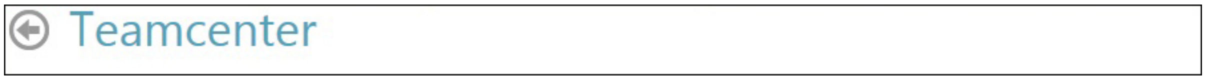
then the **url** = http://vm-tcserver28:7030/awc?embeddedLocationView=true#com.siemens.splm.clientfx.tcui.xrt.showObject;uid=gZYV3DH1qG1YeC

Settings

`Settings` define when the client should display in the embedded location view. When `embeddedLocationView` is set to **false**, the client displays as:



When `embeddedLocationView` is set to **true**, the client displays as:



Location

`Location` defines which tab elements display.

5.2.2.3 Visualization in Active Workspace

Visualization is supported with Active Workspace 4.0 by activating the **IS_VIEWER_SUPPORTED** option in the *%TCICV5_DIR%\env\aw_config.properties* file.

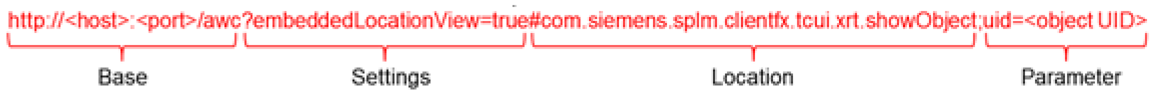
Set this option to **false** to disable visualization in Active Workspace.

```
awhosted.option.IS_VIEWER_SUPPORTED = true
```

5.2.3 Standalone Active Workspace Settings and Location

Use *%TCICV5_DIR%\env\aw_config.properties* file to configure Standalone Active Workspace settings.

The URL uses **Settings** and **Location** values to define the way the item should be opened in Active Workspace.


http://<host>:<port>/awc?embeddedLocationView=true#com.siemens.splm.clientfx.tcui.xrt.showObject;uid=<object UID>
Base
Settings
Location
Parameter

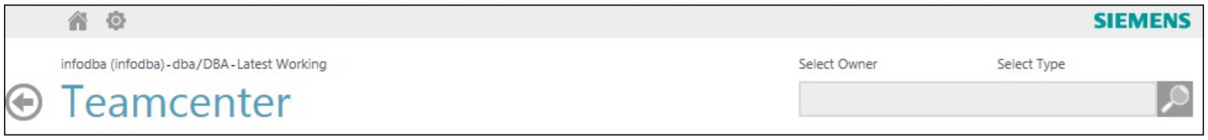
For example, when:

- **Settings**=embeddedLocationView=false
- **Location**=com.siemens.splm.clientfx.tcui.xrt.showObject;page=Overview

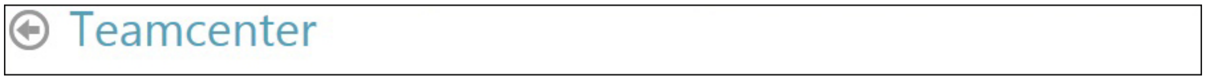
then the **url** = http://vm-tcserver28:7030/awc?embeddedLocationView=true#com.siemens.splm.clientfx.tcui.xrt.showObject;uid=gZYV3DH1qG1YeC

Settings

`Settings` define when the client should display in the embedded location view. When `embeddedLocationView` is set to **false**, the client displays as:



When `embeddedLocationView` is set to **true**, the client displays as:



Location

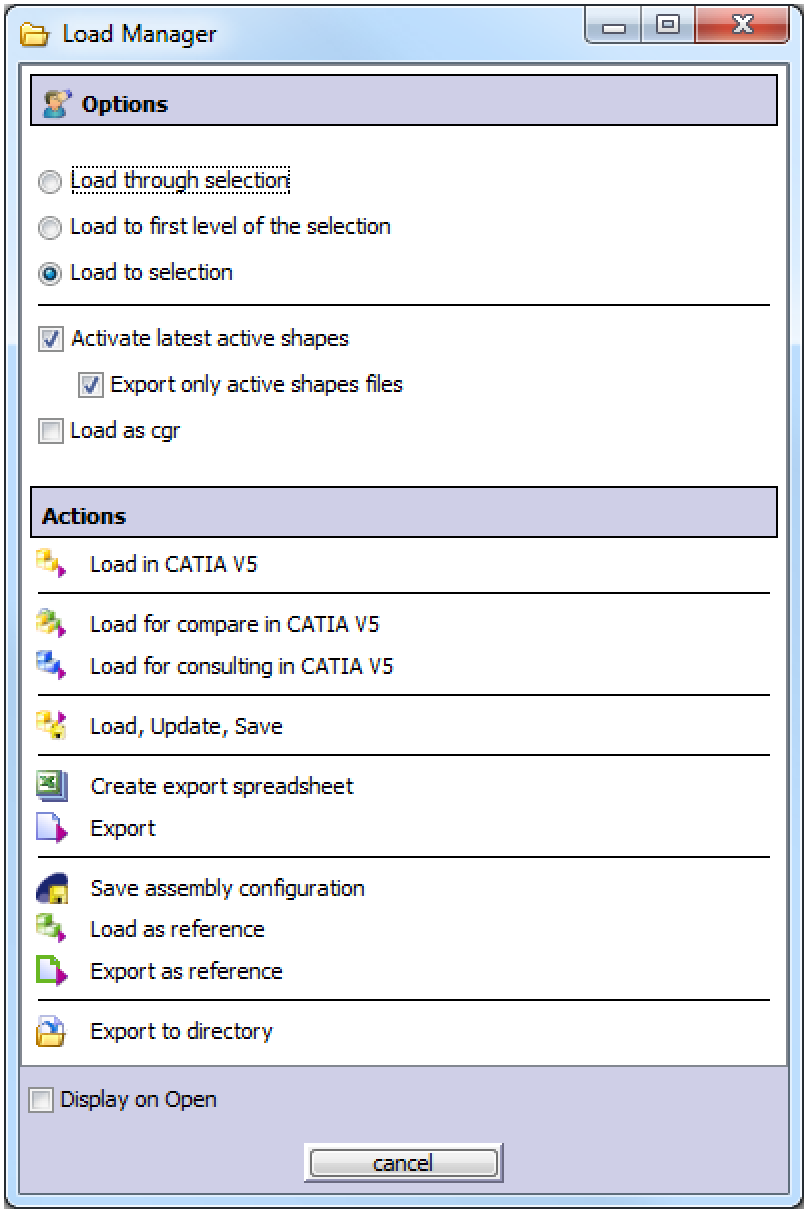
`Location` defines which tab elements display.

5.2.4 Load Manager

[Load Manager Panel](#)
[Teamcenter Loader Contextual Menu](#)
[Display structure Contextual Menu](#)

5.2.4.1 Load Manager Panel

Use the *%TCIC_DIR/env/loadmanager.xml* file provided with the installation to configure the Load Manager panel.


Load Manager

Options

- ○ Load through selection
- ○ Load to first level of the selection
- ● Load to selection

- ☑ Activate latest active shapes
  - ☑ Export only active shapes files
- ☐ Load as cgr

Actions

- Load in CATIA V5
- Load for compare in CATIA V5
- Load for consulting in CATIA V5
- Load, Update, Save
- Create export spreadsheet
- Export
- Save assembly configuration
- Load as reference
- Export as reference
- Export to directory

☐ Display on Open

By default, the file contains the following commands

```
<!-- Commands definition -->
   <Command name="load_selection"/>
   <Command name="activate_shape"/>
   <Command name="export_active_shapes"/>
   <Command name="load_as_cgr"/>
   <Command name="load"/>
   <Command name="load_compare"/>
   <Command name="load_consulting"/>
   <Command name="lus"/>
   <Command name="create_exp_spreadsheet"/>
   <Command name="export"/>
   <Command name="load_as_ref"/>
   <Command name="export_as_ref"/>
```

5.2.4.2 Teamcenter Loader Contextual Menu

Use the *%TCICV5_DIR%\env\tcicloader.xml* file to hide commands from the **Teamcenter Loader** contextual menu.

```
<Command name="paste"
image="/com/ebsolutions/uimanager/modules/images/loader/paste_16.pn
g"/>
<Command name="create_folder"/>
<Command name="remove_folder"/>
<Command name="load"
image="/com/ebsolutions/uimanager/modules/images/loader/export_32.p
ng"/>
<Command name="open_dataset"
image="/com/ebsolutions/uimanager/modules/images/loader/open_16.png
"/>
<Command name="load_compare"
image="/com/ebsolutions/uimanager/modules/images/loader/export_comp
_16.png"/>
<Command name="load_consulting"
image="/com/ebsolutions/uimanager/modules/images/loader/export_cons
ult_16.png"/>
<Command name="lus"
image="/com/ebsolutions/uimanager/modules/images/loader/lus_16.png"
/>
<Command name="create_exp_spreadsheet"
image="/com/ebsolutions/uimanager/modules/images/loader/create_spre
adsheet_16.png"/>
<Command name="export"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_16.png"/>
<Command name="load_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/export_ref_
16.png"/>
```

```
<Command name="export_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_ref_16.png"/>
<Command name="display_properties"
image="/com/ebsolutions/uimanager/modules/images/loader/properties_
32.png"/>
<Command name="display_structure"
image="/com/ebsolutions/uimanager/modules/images/loader/bomview_32.
png"/>
<Command name="load_selection"/>
<Command name="highlight"
image="/com/ebsolutions/uimanager/modules/images/loader/highlight_s
elected_16.png"/>
<Command name="show_in_teamcenter"
image="/com/ebsolutions/uimanager/modules/images/loader/sendto_16.p
ng"/>
<Command name="bmml"
image="/com/ebsolutions/uimanager/modules/images/loader/bmml_16.png
"/>
```

5.2.4.3 Display structure Contextual Menu

Use the *%TCICV5_DIR%\env\tcicloader.xml* file to hide some commands in the **Display structure** contextual menu.

```
 <Command name="open_dataset"
image="/com/ebsolutions/uimanager/modules/images/loader/open_16.png
"/>
<Command name="load_compare"
image="/com/ebsolutions/uimanager/modules/images/loader/export_comp
_16.png"/>
<Command name="load_consulting"
image="/com/ebsolutions/uimanager/modules/images/loader/export_cons
ult_16.png"/>
<Command name="lus"
image="/com/ebsolutions/uimanager/modules/images/loader/lus_16.png"
/>
<Command name="create_exp_spreadsheet"
image="/com/ebsolutions/uimanager/modules/images/loader/create_spre
adsheet_16.png"/>
<Command name="export"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_16.png"/>
<Command name="load_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/export_ref_
16.png"/>
<Command name="export_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_ref_16.png"/>
<Command name="display_properties"
image="/com/ebsolutions/uimanager/modules/images/loader/properties_
```

```
32.png"/>
<Command name="display_structure"
image="/com/ebsolutions/uimanager/modules/images/loader/bomview_32.
png"/>
<Command name="load_selection"/>
<Command name="highlight"
image="/com/ebsolutions/uimanager/modules/images/loader/highlight_s
elected_16.png"/>
<Command name="show_in_teamcenter"
image="/com/ebsolutions/uimanager/modules/images/loader/sendto_16.p
ng"/>
<Command name="bmml"
image="/com/ebsolutions/uimanager/modules/images/loader/bmml_16.png
"/>
```

5.2.5 Open in CATIA

Customize *%TCICV5_DIR%\aw\TcICLauncher.bat* file to define the behavior during the Open in CATIA command.

5.3 Configure CATIA V5

It's highly recommended to configure CATIA to store computation and result files in the same location as CATAnalysis files, by default.

5.4 Configure Teamcenter

Important Information: Database Configuration

Active Workspace

Custom Configurations

Multi-site Configurations

New Instance Synchronization

Templates

Preferences Related to the CATIA V5 Menu in Teamcenter

Important Information: Database Configuration

Pick **Templates Extraction** from the InstallAnywhere panel to retrieve the template required for database configuration.

Start the Teamcenter Environment Manager, based on your system below, to access **templates** and configure the database schema.

Operating System	File
Linux	TC_ROOT/install/TEM.sh
UNIX	TC_ROOT/install/TEM.sh
Windows	TC_ROOT\install\TEM.bat

⚠ ***You must be logged in as a Database Administrator to complete the configuration process using the TEM files.***

5.4.1 Active Workspace

⚠ ***The Active Workspace MCAD Integration template must be deployed prior to using TcIC with Active Workspace.***

5.4.2 Custom Configurations

It is recommended to adopt the standard Teamcenter Integration for CATIA configuration, however, alternate behavior may be required.

Teamcenter Integration for CATIA behavior may be modified using **preferences** defined in Teamcenter. For example, it is possible to define file naming conventions, BOM view type or default mapping used by the Save Manager. Preferences that are undefined adopt their default documented value.

Individual preferences may be defined using the Teamcenter Options dialog. It is also possible to import a collection of preferences to apply specific configuration requirements.

Before attempting to import preferences, it is strongly recommended to review all preference values to ensure accuracy and save the preferences as your site integration preferences file.

Perform the following steps to import a preferences .xml file.

1. Pick **Edit >> Options**.
2. Pick **Index**.
3. Pick **Import** from the right panel.
4. Pick the .xml preferences file.
5. Pick **Site** for the scope and **General** for the category. Site preferences apply to all users.
6. Merge the values of a given preference, if they already exist.

5.4.3 Multi-site Configurations

The Integration uses user-data attached to Integration relations. Set **TC_always_export_user_data_for_relation_type** to include Integration relations and avoid data inconsistencies caused when Integration relations are not transversed in multi-site configurations (resulting in *stubs* at the transfer site).

Use TC_always_export_user_data_for_relation_type to ensure the user data is **always** exported with GRM relations, even if the secondary object is excluded. The user data object will always have the same site ownership as the GRM relation itself.

```
TC_always_export_user_data_for_relatio
n_type=

catiaV5_DWGLink
catiaV5_MML
catiaV5_MMLAssy
catia_alternateShapeRep
catia_componentShapeRep
catia_MFGLink
catia_analysisLink
catia_analysisFilesLink
catia_cache_link
catia_secondary_link
```

5.4.4 New Instance Synchronization

The **PSEOccNotesNoCopy** setting contains the following values after successful installation.

- catiaOccurrenceName
- catiaParentPartName
- catiaFileName
- catiaOccurrenceOEM
- catiaOccurrenceDesc

These values are checked when the **New Instance Synchronization** process begins during a **Save Existing** process. The process stops and an error message displays when one or more of these values are missing in the database. The error message specifies which value(s) is missing.

5.5 Templates

5.5.1 Template Prerequisites

Conflicts may occur when customizations have been previously applied to the Integration. Run the Teamcenter TypeAnalysisTool before applying a new Teamcenter Integration for CATIA V5 template to ensure type collisions will not occur.

Refer to Siemens' Teamcenter documentation for information on Template collision check and contact GTAC for support with these errors.

5.5.2 Accessing the Templates

Perform the following steps to access templates provided with the connector (`TCICV5_DIR/install/template`).

1. Pick **Configuration Manager** from the Maintenance panel, then pick **Next**.
2. Pick **Perform maintenance on an existing configuration** then pick **Next**.
3. Pick the configuration from which the corporate server was installed from the Configuration Selection pane.
4. Pick **Next**.
5. Pick **Add/Remove Features** under the Teamcenter section of the Feature Maintenance panel.
6. Pick the **Browse** button from the Select Features panel.
7. Pick **All Files** from the browsing dialog box to view all the files in the system.
8. Browse to the template files directory, i.e., *TCICV5_DIR/install/template* for Windows, or browse to the directory containing the extracted template.
9. Ensure **Feature Files** is selected to view only the template (feature) files available for installation.
10. Pick the required template's feature file (feature_integration4catia.xml).
11. Pick **Select**.
12. Pick the new template just chosen from the Select Features panel under Teamcenter Corporate Server (it is listed within the Selection list).
13. Pick **Next**.
14. Enter a username and password in the Teamcenter Administrative User Panel to log on to the server.
15. Pick **Next**.
16. The Database Template Summary panel displays a list of installed templates. Pick **Next**.
17. Pick **Next** from the Confirm Selections panel to confirm the selection and install the new template.

5.5.3 Display Specific Relations

Beginning with **Teamcenter 8**, a combination of Teamcenter preferences and BMIDE configuration (using a template) is required in order to enable the display and navigation of objects using specific relation types in MyTeamcenter.

- BMIDE configuration is used to define specific relations which may later display at objects in Teamcenter, such as ItemRevisions and datasets.
- Teamcenter preferences are used to enable the display of these related objects using MyTeamcenter.

Prior to Teamcenter 8, both components were achieved using Teamcenter preferences. In addition, configuration required by the Integration for **Multi-model Link Browsing** occurred during the installation, but enabling the display or extending definitions to support additional relations such as *cache* or *derived* datasets were the result of specific user-defined customizations.

Beginning with **Teamcenter Integration for CATIA V5 (8.2.0)**, the Integration performs the necessary BMIDE configuration; it is only necessary to define Teamcenter preferences to enable the display of specific relations.

Refer to the following table for important information regarding this process.

If	And	Result	Fix
a custom template is used to apply a BMIDE configuration	TEM is used to apply Teamcenter Integration for CATIA V5, versions 8.2.0 or later, during an update or upgrade	The update or upgrade fails due to detected collisions between the custom template and the Teamcenter Integration for CATIA V5 template, versions 8.2.0 and later. *This is expected behavior.*	The custom template should not be applied or, if applied, the conflicting BMIDE definitions must be removed from the custom template.
an **update** is performed *(apply changes to existing templates without changing the Teamcenter version)*.	the updated template and the Integration template (optional) compatible with the current Teamcenter release is applied.		

an **upgrade** is performed *(replace one Teamcenter version with another)*	*Required*: the updated customer template and the Integration template compatible with the current Teamcenter release must be applied.		

5.5.4 Preferences Related to the CATIA V5 Menu in Teamcenter

Teamcenter must be restarted after modifying preferences.

5.5.4.1 Create Dataset Items Using the PSE Menu

When the I-deas NX Manager is active or the necessary libraries (available from Siemens for Windows, Solaris and HP-UX) have been separately configured and installed, additional options under the CATIA menu of the PSE can be enabled.

These options can be used to create CATPart and CATProduct dataset(s) for selected item revisions, if they do not already exist.

Enable these options by setting **IMAN_show_open_in_ideas_button** within Teamcenter Preferences.

```
IMAN_show_open_in_ideas_button = true
```

The default value is **true**.

5.6 Configure Teamcenter Loader

5.6.1 Teamcenter Preferences

Features of the Display structure panel are controlled by the following Teamcenter preferences.

Feature	Teamcenter Preference and Description
Table Column	**PortalPSEColumnsShownPref** :Specifies the list of PSE Columns displayed (left-to-right order). To display the Explore Checked Out property, add **bl_rev_expl_checkout** to this list Default value: List of BOM Lines attributes.
Table Column Size	**PortalPSEShownColumnWidthsPref**: Specifies the widths for the PSE shown Columns (left-to-right order). Default value: List of column sizes.
Display	**PSEAutoPackPref**: Indicates if the PSE display needs to be packed by default. The default value is 1. **PSEImpreciseColorPref**: Specifies the color to be used when displaying an imprecise BVR in PSE (defaulted to white). The value can be a standard supported string, an octal number or a hexadecimal number. The standard strings supported are black, white, lightGray, gray, darkGray, red, pink, orange, yellow, green, magenta, cyan and blue. Example for a hexadecimal number is #50b080 which represents Sea Green. The default value is white. **PSEPreciseColorPref**: Specifies the color to be used when displaying a precise BVR in PSE (defaulted to sea green). The value can be a standard supported string, an octal number or a hexadecimal number. The standard strings supported are black, white, lightGray, gray, darkGray, red, pink, orange, yellow, green, magenta, cyan and blue. Example for a hexadecimal number is #50b080 which represents Sea Green. The default value is #50b080. **PSE_expand_on_open**: Indicates if an object's root node is expanded when displayed in Structure Manager. The default value is 1.

5.7 Configure Teamcenter Integration for CATIA V5

Attribute Mappings
CGR Management
Ignore Nodes
Import Queries
Integration Schema
Integration Toolbar
Multiple CATIA Environments
Theorem/Teamcenter Translator (JT)

See Also:

View Configuration Details

5.7.1 Attribute Mappings

Behavior and Modifying CATIA Properties
Configuring Attribute Mappings
For CATDrawings, CATParts, and CATProducts
For Design Tables
For Foreign Files
For the Integration Schema
For the Save Manager
Preferences Related to Attribute Mappings

Define Attribute Mappings between CATIA and Teamcenter properties. Attribute Mappings are:

- a convenient way to link properties from CAD with properties in Teamcenter
- available for the Save, Load, Import / Export, and Update Title Block processes of Teamcenter Integration for CATIA V5.
- required for Import / Export processes. They are used to store OEM attributes of a CATIA V5 component in their corresponding form in Teamcenter during the Import process.
- used to restore OEM attributes in CATIA V5 components during the Export process.
- can be defined between a CATIA drawing text or drawing parameters and Teamcenter properties.

Refer to Teamcenter's Configuration Guide for supplemental information.

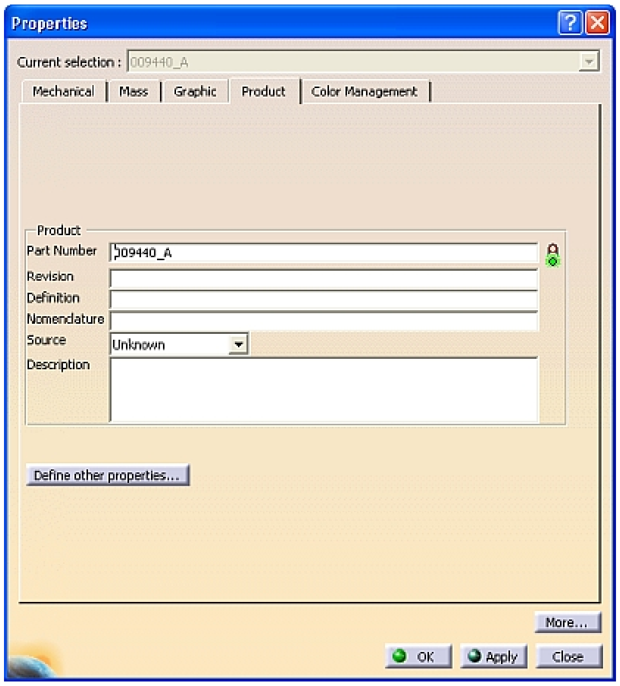
5.7.1.1 Behavior and Modifying CATIA Properties

The **Definition** and **Revision** CATIA properties are not mapped to the Item_ID and ItemRev_ID and the EBS_DatasetUID, EBS_DatasetLastVersion, and EBS_ItemRevName properties are not created.

This behavior applies to Product and Part properties and Drawing parameters. Information in the *.dat* file is used to create a correspondence between CATIA documents and Teamcenter datasets. Information in CATIA properties is not used.

Mappings between Revision and Itemrev_ID and Definition and Item_ID do not exist.

The EBS_DatasetUID, EBS_DatasetLastVersion, and EBS_ItemRevName properties are not created.



Part Number	009440_A
Revision	
Definition	
Nomenclature	
Source	Unknown
Description	

5.7.1.1.1 Modify the Behavior of CATIA Properties

Definition/Revision Properties

Mappings between the **Definition** CATIA property and Item_ID and **Revision** CATIA property and ItemRev_ID can be defined as follows:

```
revision : ItemRevision.item_revision_id/master-iman
/description="Dokument Name"
Definition : ItemRevision.item_id /master-iman
/description="Dokument Name"
```

Reference Properties

The EBS_DatasetUid and EBS_DatasetVersion properties are never created even if a mapping is defined.

Define Teamcenter mappings on the EBS_ItemRevName reference property as follows:

```
EBS_ItemRevName : ItemRevision.object_name /master-iman
/description="Dokument Name"
```

Mappings between EBS_DatasetUID and Teamcenter dataset_uid is not permitted because this information is not accessible to Teamcenter mappings.

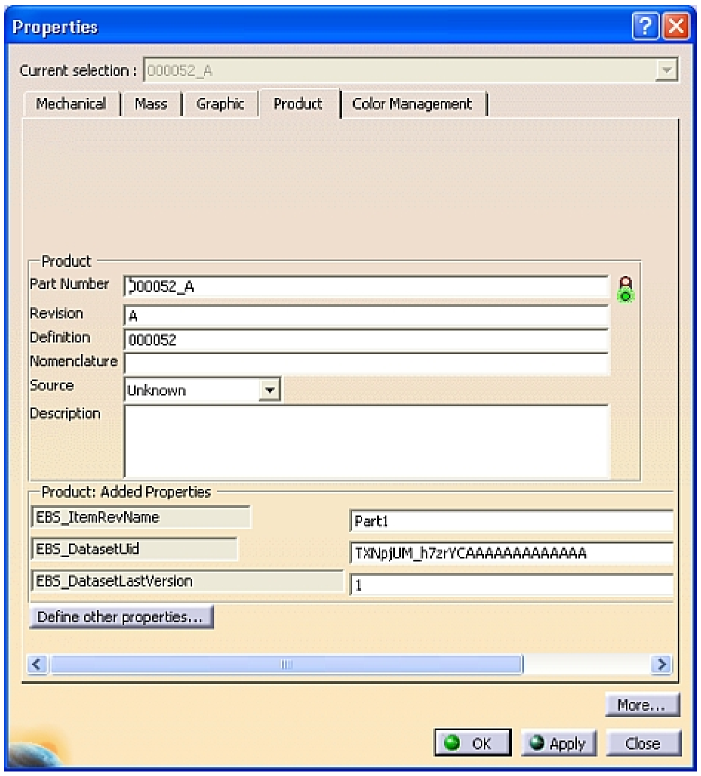
Mappings between EBS_DatasetLastVersion and Teamcenter dataset_version is not permitted because this value is updated after importing the file to the database. If permitted, this would cause the CATIA property value to be modified after performing an update of CATIA properties and/or after the Load process.

Revision = ItemRev_ID
Definition = Item_ID

EBS properties are created:

- EBS_DatasetUid = dataset_uid
- EBS_DatasetVersion = dataset_version

- EBS_ItemRevName = Itemrev_Name



Part Number	000052_A
Revision	A
Definition	000052
Nomenclature	
Source	Unknown
Description	
EBS_ItemRevName	Part1
EBS_DatasetUid	TXNpjUM_h7zrYCAAAAAAAAAAAAA
EBS_DatasetLastVersion	1

5.7.1.2 Configuring Attribute Mappings

[Determine the Real Name of a Form's Properties](#)

Use caution when configuring Attribute Mappings to avoid potential status inconsistencies on parts, products, and/or drawings between Teamcenter and CATIA, caused by incompatible mapping definitions.

Refer to Teamcenter's Configuration Guide for supplemental information.

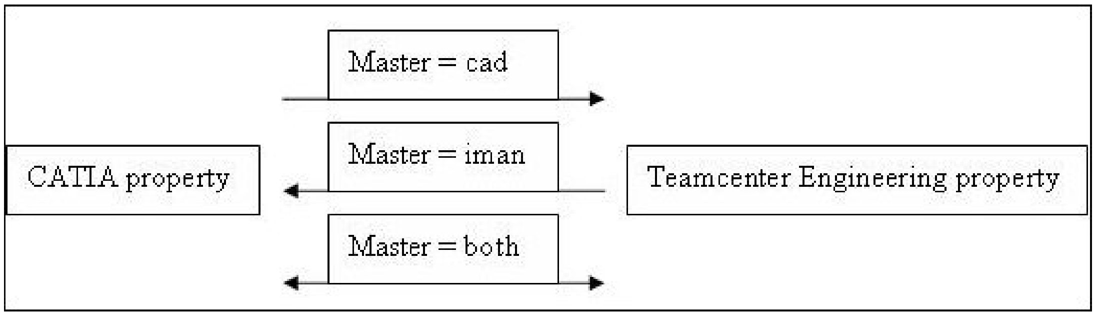
Important Notes

- Attribute Mappings are case-sensitive.
- Define which value takes precedence or prevent a difference from occurring at all.

- Properties must be created in the CATIA component prior to defining an attribute mapping on a new CATIA property. They cannot be mapped to non-existing property.
- When **createPropForMapping** is **True** and new properties (non-standard) are found, then the new properties are created during the attribute mapping process.

Perform the following steps to define attribute mappings.

1. **Export** the existing mappings to a text file.
2. **Define** the mappings.
3. **Import** the modified file into the database. When the property is created/updated in CATIA, the prefix is removed and the property is created per its type. New properties are created during the attribute mapping process when the **createPropForMapping** setting is **True** and new properties (non-standard) are found.

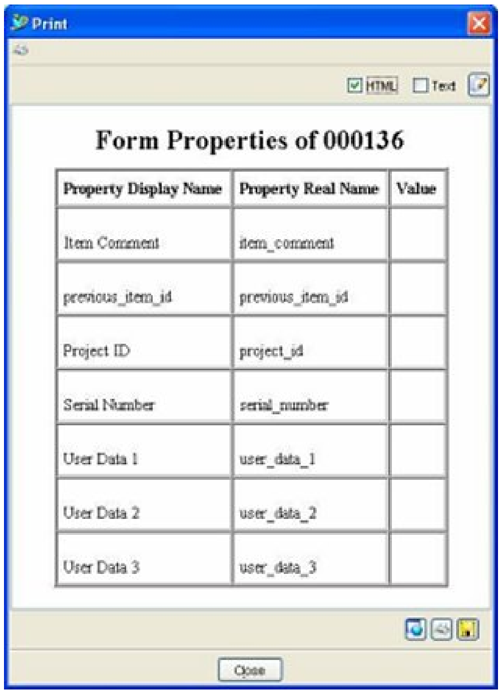


If	Then
`master = cad`	mappings are not available on `item ID` and `item revision ID` of Teamcenter
master = iman for a CATIA component	mappings are not available on `EBS_DatasetLastVersion, EBS_BvrLastModifiedDate, Revision` and `Definition` CATIA properties
master = iman, for a drawing	mappings are not available on `DatasetUID, ItemID, ItemRevID, catia_type_value` and `DatasetLastVersion` CATIA properties

5.7.1.2.1 Determine the Real Name of a Form's Properties

The Real Names of Teamcenter properties must be used. For example, perform the following steps to determine the real name of a form's properties.

1. **Open** the form.
2. Pick **Print the file**.
3. Pick the **Set result format** button, then pick the **Real Name** option.



Form Properties of 000136

Property Display Name	Property Real Name	Value
Item Comment	item_comment	
previous_item_id	previous_item_id	
Project ID	project_id	
Serial Number	serial_number	
User Data 1	user_data_1	
User Data 2	user_data_2	
User Data 3	user_data_3	

The same steps may be performed from within the properties panel.

Property Name displayed in the Properties Panel in CATIA	Property Name to use for mapping
Part Number	partNumber

Property Name displayed in the Properties Panel in CATIA	Property Name to use for mapping
Revision	revision
Definition	definition
Nomenclature	nomenclature
Source	Source
Description (in the Product panel)	description
Description (in the Component panel)	DescriptionInst
MyProperty (new property created in CATIA)	MyProperty

The CATIA parameters names list for a drawing :

- `DatasetUID`
- `ItemID`
- `ItemRevID`
- `ItemRev_Name`
- `catia_type_value`
- `DatasetLastVersion`

5.7.1.3 Configuring Attribute Mappings for CATDrawings, CATParts and CATProducts

CATDrawing Parameters and Text within a CATDrawing

CATPart Example

Attribute mappings are required for CATDrawing, CATPart and CATProduct, and must be configured for the supplier import process.

Refer to the Export_Mappings.txt file located in the *TcIC_DIR\sample* folder for detailed configuration information.

5.7.1.3.1 CATDrawing Parameters and Text within a CATDrawing

Attribute Mappings can be defined between a CATDrawing text or drawing parameters and Teamcenter properties.

When a CATDrawing parameter is linked to a title block with a CATIA formula, the modified parameter value is propagated to the title block. However, when the title block is modified, it is not propagated back to the parameter.

Perform the following steps to define the attribute map.

1. Ensure an attribute map has been created for a CATDrawing. For example:

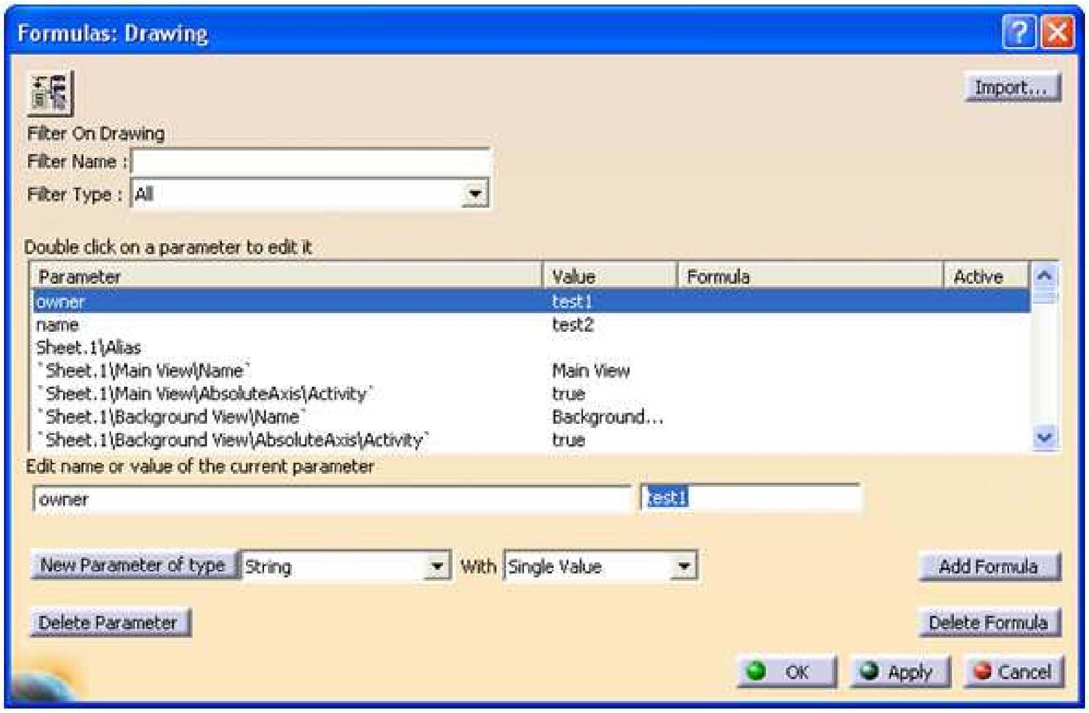
```
{ Dataset type="CATDrawing"
     { Item type="item"
          owner : owning_user.user_name /description="owning user"
          name
:  ItemRevision.object_name/master=cad/desciption="ItemRevision
name"
     }
}
```

2. Pick **Formula** f(x) from the Standard toolbar to create a new parameter with the name of the map.

3. Select **New Parameter of type** from the Formulas: Drawing dialog box.

4. Complete the **Edit name or value of the current parameter** fields. For example, *owner* and *test1.*

5. Repeat the previous steps for the parameter name, then pick **OK** when finished.



Double click on a parameter to edit it

Parameter	Value	Formula	Active
owner	test1		
name	test2		
Sheet.1\Alias			
`Sheet.1\Main View\Name`	Main View		
`Sheet.1\Main View\AbsoluteAxis\Activity`	true		
`Sheet.1\Background View\Name`	Background...		
`Sheet.1\Background View\AbsoluteAxis\Activity`	true		

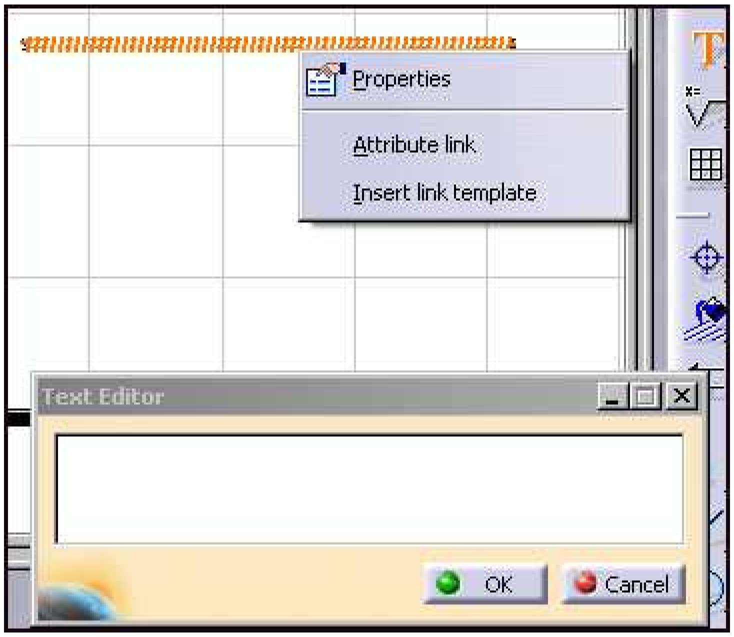
Edit name or value of the current parameter

owner test1

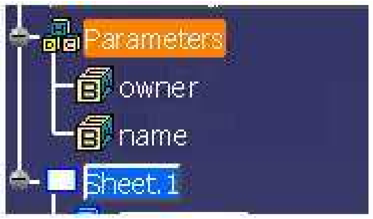
New Parameter of type: String With: Single Value

6. Add text to the drawing. Do not close the text editor.

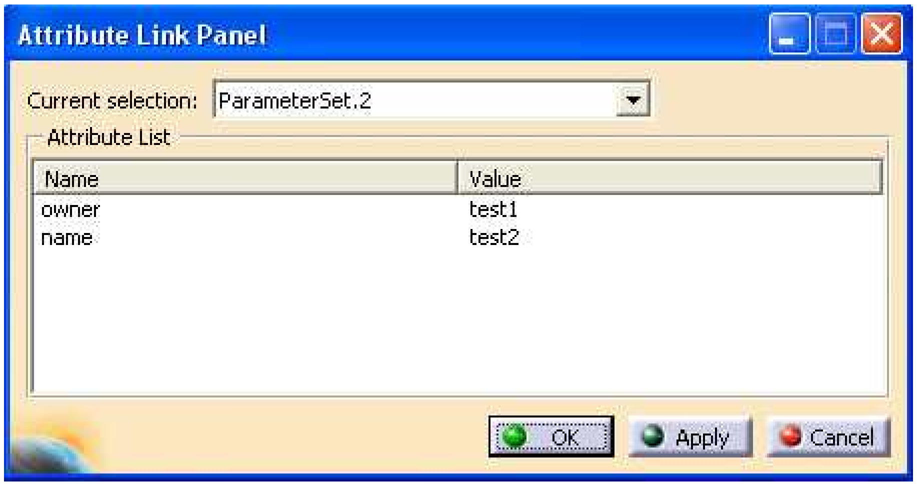
7. Click on the green text border then use the right-mouse button and pick **Attribute link** to link the parameter to the text box.



8. Select **Parameters** from the specification tree.



9. Select the attribute name just created from the Attribute Link Panel, then pick **OK**. Repeat the previous steps to link the parameter name.



Current selection: ParameterSet.2

Attribute List

Name	Value
owner	test1
name	test2

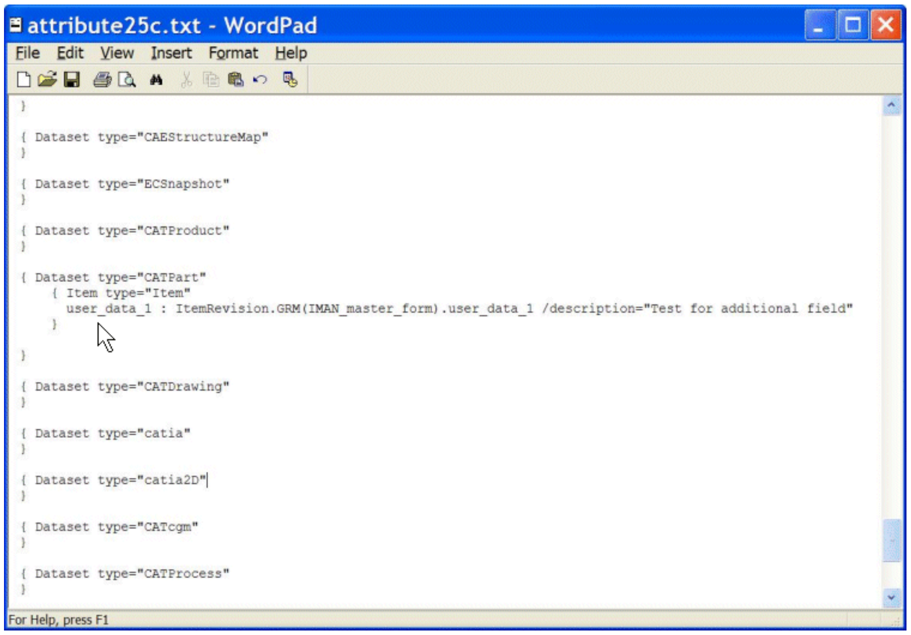
10. Pick the **Update Title Block** button from the Integration toolbar. The first text updates with attribute values of the owning user of the CATDrawing and the second text updates with the attribute values of the CATDrawing item revision name.

5.7.1.3.2 *CATPart Example*

Perform the following steps to create a custom attribute, *user_data_1*, attached to a CATPart.

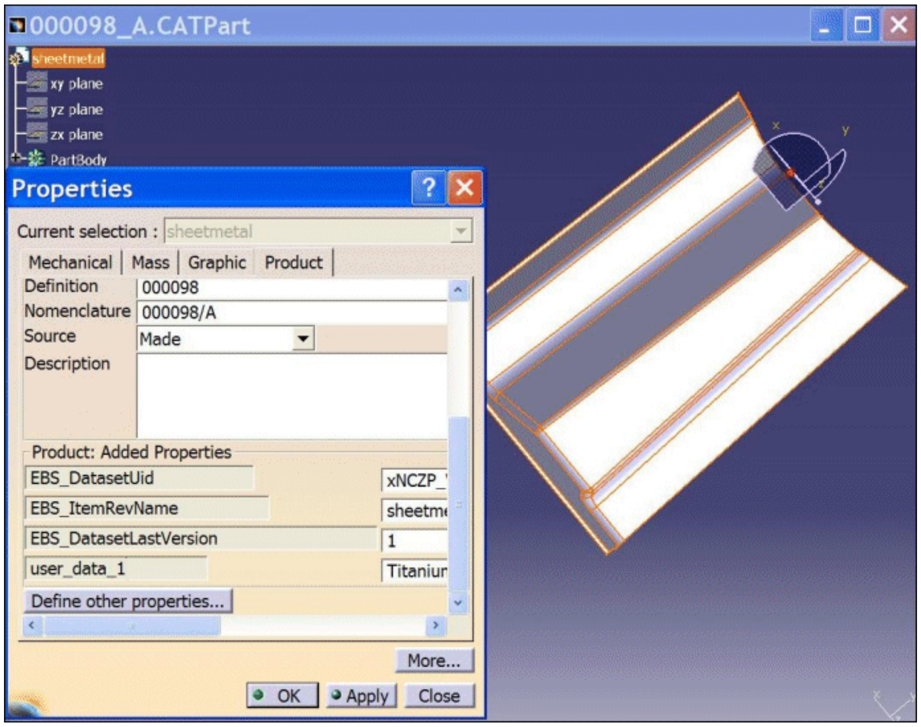
1. Export the attribute map.

2. Modify the attribute map by adding the following line within the attribute map, as shown: `user_data_1: ItemRevision.GRM(IMAN_master_form).user_data_1`.



```
}

{ Dataset type="CAEStructureMap"
}

{ Dataset type="ECSnapshot"
}

{ Dataset type="CATProduct"
}

{ Dataset type="CATPart"
    { Item type="Item"
      user_data_1 : ItemRevision.GRM(IMAN_master_form).user_data_1 /description="Test for additional field"
    }

}

{ Dataset type="CATDrawing"
}

{ Dataset type="catia"
}

{ Dataset type="catia2D"
}

{ Dataset type="CATcgm"
}

{ Dataset type="CATProcess"
}
```

3. **Save** the attribute map file, then **Import** it. The user_data_1 field has been added to the properties panel.



Current selection : sheetmetal

Mechanical | Mass | Graphic | Product

Definition	000098
Nomenclature	000098/A
Source	Made
Description	

Product: Added Properties

EBS_DatasetUid	xNCZP_
EBS_ItemRevName	sheetm
EBS_DatasetLastVersion	1
user_data_1	Titaniur

Define other properties...

5.7.1.4 Configuring Attribute Mappings for DesignTables

Ensure the correct mappings have been added to the Attribute Mappings file. The same mappings are used for Excel and Text dataset types.

For example:

```
{ Dataset type-"MSExcel"
    { Item type-"DesignTable"
          OEM_partNumber : NR(catimportform).OEM_partNumber
description-"Mapping Test"
          OEM_filePath : NR(catimportform).OEM_filePath
/description-"Mapping Test"
          OEM_definition : NR(catimportform).OEM_definition
/description-"Mapping Test"
    }
    { Item type-"Item"
          OEM_partNumber : NR(catimportform).OEM_partNumber
/description-"Mapping Test"
          OEM_filePath : NR(catimportform).OEM_filePath
/description-"Mapping Test"
          OEM_definition : NR(catimportform).OEM_definition
/description-"Mapping Test"
```

```
    }
}
{ Dataset type-"Text"
    { Item type-"DesignTable"
          OEM_partNumber : NR(catimportform).OEM_partNumber
description-"Mapping Test"
          OEM_filePath : NR(catimportform).OEM_filePath
/description-"Mapping Test"
          OEM_definition : NR(catimportform).OEM_definition
/description-"Mapping Test"
}
{ Item type-"Item"
       OEM_partNumber : NR(catimportform).OEM_partNumber
/description-"Mapping Test"
       OEM_filePath : NR(catimportform).OEM_filePath /description-
"Mapping Test"
       OEM_definition : NR(catimportform).OEM_definition
/description-"Mapping Test"
}
```

5.7.1.5 Configuring Attribute Mappings for Foreign Files

Refer to the following behavior of foreign files when **CATIA_manage_foreignfiles_on_save** is **true**.

Mapping must be added to locate a foreign file in the dataset. A mapping must be added for each dataset type listed in the **CATIA_component_dataset_types** preference.

Using the dataset catia causes the mapping process to add because catia is listed in the CATIA_component_dataset_types preference by default, even if the dataset is not visible in the preference. Mappings must be set when the catia is used.

Refer to the following example to add mappings for CATIA_component_dataset_types = CATwrl, CATShape.

```
{ Dataset type-"catia"
   { Item type-"Item"
       OEM_revision : GRM(catia_importform_link).OEM_revision
/description-"Mapping Test"
        OEM_partNumber : GRM(catia_importform_link).OEM_partNumber
/description-"Mapping Test"
        OEM_nomenclature :
GRM(catia_importform_link).OEM_nomenclature /description-"Mapping
Test"
```

```
        OEM_filePath : GRM(catia_importform_link).OEM_filePath
/description-"Mapping Test"
        OEM_dlname : GRM(catia_importform_link).OEM_dlname
/description-"Mapping Test"
        OEM_description :
GRM(catia_importform_link).OEM_description /description-"Mapping
Test"
        OEM_definition : GRM(catia_importform_link).OEM_definition
/description-"Mapping Test"
   }
}
{ Dataset type-"CATwrl"
    { Item type-"Item"
        OEM_revision : GRM(catia_importform_link).OEM_revision
/description-"Mapping Test"
        OEM_partNumber : GRM(catia_importform_link).OEM_partNumber
/description-"Mapping Test"
        OEM_nomenclature :
GRM(catia_importform_link).OEM_nomenclature /description-"Mapping
Test"
        OEM_filePath : GRM(catia_importform_link).OEM_filePath
/description-"Mapping Test"
        OEM_dlname : GRM(catia_importform_link).OEM_dlname
/description-"Mapping Test"
        OEM_description :
GRM(catia_importform_link).OEM_description /description-"Mapping
Test"
        OEM_definition : GRM(catia_importform_link).OEM_definition
/description-"Mapping Test"
    }
}
{ Dataset type-"CATShape"
    { Item type-"Item"
        OEM_revision : GRM(catia_importform_link).OEM_revision
/description-"Mapping Test"
        OEM_partNumber : GRM(catia_importform_link).OEM_partNumber
/description-"Mapping Test"
        OEM_nomenclature :
GRM(catia_importform_link).OEM_nomenclature /description-"Mapping
Test"
        OEM_filePath : GRM(catia_importform_link).OEM_filePath
/description-"Mapping Test"
        OEM_dlname : GRM(catia_importform_link).OEM_dlname
/description-"Mapping Test"
        OEM_description :
GRM(catia_importform_link).OEM_description /description-"Mapping
Test"
        OEM_definition : GRM(catia_importform_link).OEM_definition
/description-"Mapping Test"
    }
}
```

5.7.1.6 Configuring Attribute Mappings for the Integration Schema

Perform the following steps to set Attributes Mappings for Import / Export management.

1. Export the existing mappings to a .txt file using the `export_attr_mappings` utility. Refer to the sample *export_mappings.txt* file in the *TcIC_DIR\sample\AttributesMappings* folder, if necessary.

2. Edit the file by adding, removing or modifying mappings. Refer to the sample *attribute_mappings.txt* file in the *TcIC_DIR\sample\AttributesMappings* folder, if necessary.

3. Import the mapping file, using the `import_attr_mappings` utility. Refer to the sample *import_mappings.txt* file in the *TcIC_DIR\sample\AttributesMappings* folder, if necessary.

Add lines within the attribute_mappings.txt file, in the TcIC_DIR\sample\AttributesMappings folder, for CATProduct, CATPart and CATDrawing dataset types in your mappings to use the Import/Export processes.

5.7.1.7 Configuring Attribute Mappings for the Save Manager

Save Manager attributes mappings are **not** case sensitive.

CATIA properties can only be mapped to Save Manager properties individually, **one at a time**.

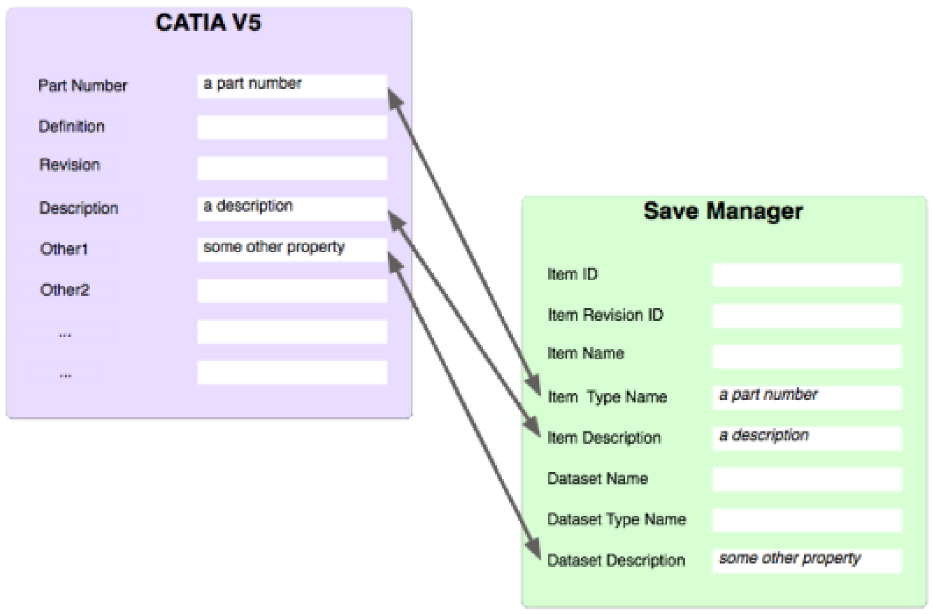
Save Manager attributes mappings are resolved only during the following scenarios:

- Create New components in CATIA
- The save mode is changed to Create New or Save As New. Save Manager attribute mappings are not applied when the save mode is change to Save As Copy.
- The Clear/Erase button is selected and the save mode is changed to Create New.
- The item type is changed in the Save Manager panel.

5.7.1.7.1 Customize Attribute Mappings

Customize the Save Manager Attribute Mappings to automatically complete fields in the Save Manager dialog for new items. For example, use Attribute Mappings to automatically add the CATIA Part Number to the Item Type Name field in the Save Manager.

Any CATIA property may be used as a mapping in the Save Manager, including custom properties. The CATIA filename property can be mapped to an item revision attribute or a dataset attribute in the Save Manager panel.



Use any combination of the following variables in Teamcenter Preferences to map a CATIA property to a Save Manager property.

- **CATIA_savemanager_map_ir_attr**: used to map a CATIA property to an item revision attribute in the Save Manager panel.
- **CATIA_savemanager_map_ds_attr**: used to map a CATIA property to a dataset attribute in the Save Manager panel.

The prefix, **attr**, may be applied to an item revision or dataset real name property as shown below.

- CATIA_savemanager_map_ir_**item_id**
- CATIA_savemanager_map_ir_item_**revision_id**
- CATIA_savemanager_map_ir_**object_name**
- CATIA_savemanager_map_ir_**object_type**
- CATIA_savemanager_map_ir_**object_desc**

- CATIA_savemanager_map_ds_**pubr_object_id**
- CATIA_savemanager_map_ds_**rev**
- CATIA_savemanager_map_ds_**object_name**
- CATIA_savemanager_map_ds_**object_type**
- CATIA_savemanager_map_ds_**object_desc**

Any CATIA property may be used, including custom properties, as a mapping in the Save Manager.

```
CATIA_savemanager_map_ir_item_id =
partNumber

CATIA_savemanager_map_ds_object_name=
description

CATIA_savemanager_map_ds_object_desc=
custom_CATIA_property
```

5.7.1.7.2 Modify Behavior

The following Save Manager Attributes must be modified by the Administrator to change their behavior.

CATIA Property	Save Manager Property
definition	Item ID
revision	Revision
partNumber	Revision Name

5.7.1.8 Preferences Related to Attribute Mappings

Create Non-standard CATIA Properties
Nomenclature and PartNumber Properties in CATIA
Manage Mapping on Instance Properties
How do I modify preferences and behaviors?

Teamcenter must be restarted after modifying preferences.

5.7.1.8.1 Create Non-standard CATIA Properties

Set **CATIA_create_catia_properties_for_mapping** to **true** to create non-standard CATIA properties corresponding to mappings if they are found during the Save and Load processes. Set it to **false** to prevent non-standard CATIA properties corresponding to mappings from being created, even if they are found.

```
CATIA_create_catia_properties_for_mapping = false
```

The default value is **false**.

This preference corresponds to the previous *com.ebsolutions.catia.createPropForMapping* property defined in the catiaV5properties.properties file, now obsolete.

5.7.1.8.2 Nomenclature and PartNumber Properties in CATIA

Declare two Teamcenter properties to define the **Nomenclature** and **PartNumber** properties in CATIA by setting **CATIA_MAP_Property_Nomenclature** and **CATIA_MAP_Property_PartNumber**. A blank line marks the end of the definition.

Multiple values are permitted.

```
CATIA_MAP_Property_Nomenclature=
Dataset.object_name
Dataset.creation_date

CATIA_MAP_Property_PartNumber=
Item.object_name
Dataset.object_type
```

In this example, dataset name (`Dataset.object_name`) and the dataset creation date (`Dataset.creation_date`) define a value of the Nomenclature property in CATIA and item name (`Item.object_name`)define a value of the PartNumber property in CATIA.

Property values are separated by the value of the CATIA_partnumber_separator Teamcenter Preference.

The required format for Teamcenter preferences values is *ClassName.RealNameProperty*.

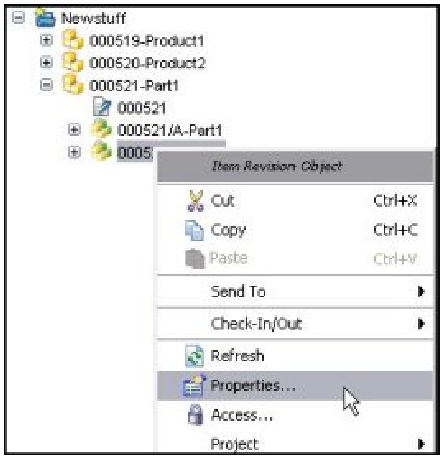
The RealNameProperty corresponds to the Teamcenter **Property Real Name** and **ClassName** corresponds to values shown in the table:

	ClassName
Item	Item
Item Revision	ItemRevision
Dataset	Dataset
BOMView Revision	PSBOMViewRevision
Item Master	ItemMaster
ItemRevisionMaster	ItemVersionMaster

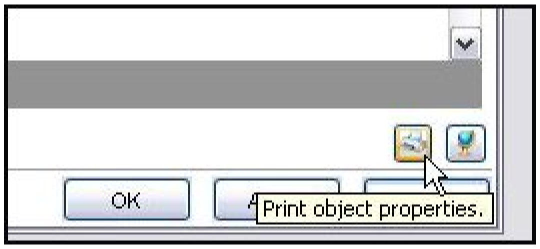
Perform the following steps to display properties:

1. Pick an item or item revision.

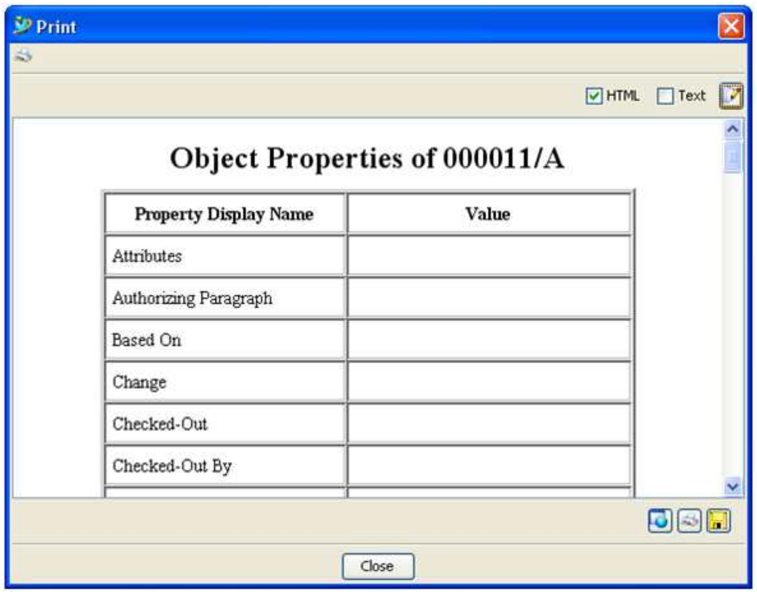
2. Right click then pick **Properties...** from the pop-up menu.



3. Pick the **Print object properties** button from the Properties form.



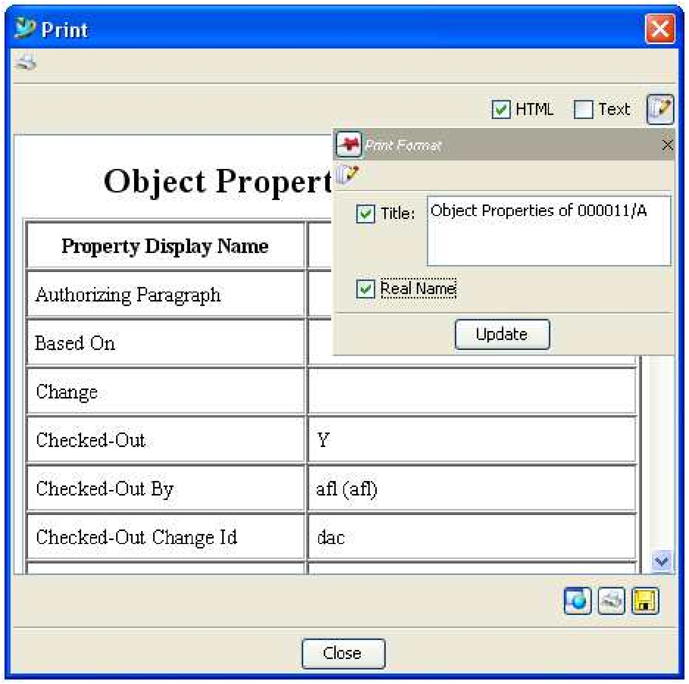
4. The **Property Display Name** column displays in the Object Properties form. Pick the **Set Result Format** button .



Object Properties of 000011/A

Property Display Name	Value
Attributes	
Authorizing Paragraph	
Based On	
Change	
Checked-Out	
Checked-Out By	

5. Toggle **Real Name** on from the Print Format form to display the Property Real Name column, then pick **Update**.

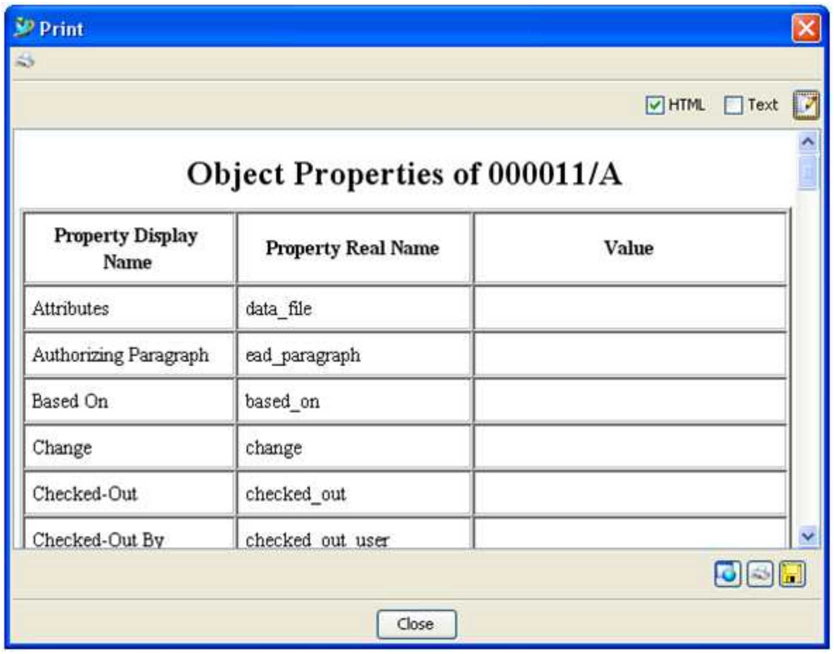


Title: Object Properties of 000011/A

☑ Real Name

Property Display Name	
Authorizing Paragraph	
Based On	
Change	
Checked-Out	Y
Checked-Out By	afl (afl)
Checked-Out Change Id	dac

6. The Property Real Name column displays in the Object Properties form. Pick one or more names in the attribute list to define, as needed.



Object Properties of 000011/A

Property Display Name	Property Real Name	Value
Attributes	data_file	
Authorizing Paragraph	ead_paragraph	
Based On	based_on	
Change	change	
Checked-Out	checked_out	
Checked-Out By	checked_out_user	

5.7.1.8.3 Multi-Field Key Functionality/Part Number Conflicts

Ensure **CATIA_MAP_Property_PartNumber** is set to **Item.multifieldkey** to avoid part number conflicts when managing multi-field key functionality. Attribute values of multi-field key can be defined for an item's class. Attributes values are ordered as it is defined in the multi-field key.

```
CATIA_partnumber_separator = _
CATIA_MAP_Property_PartNumber = Item.multifieldkey,
ItemRevision.item_revision_id
CATIA_MAP_Property_Nomenclature = Item.multifieldkey
```

For example, an item is Cat2ItemMFK10{item_id,cat2str_3,cat2str4}. The values: item_id = *id1*, cat2str_3 = *value1*, cat2str_4 = *value2*, item_revision_id = *A* results in PartNumber value = *id1_value1_value2_A* and the Nomenclature value is *id1_value1_value2.*

5.7.1.8.4 Manage Mapping on Instance Properties

Checksum

Set **CATIA_occmapp_checksum_note** to define the occurrence note type used to store the checksum of mappings on instances. Checksum information determines whether the mapping should be reapplied.

```
CATIA_occmapp_checksum_note = Cat2OccMapCksum
```

The default value is **Cat2OccMapCksum**.

5.7.1.8.5 List of Occurrences Notes

Set **CATIA_occmap_tcproperties** to define the list of all occurrences notes and BOMLine attributes used for mappings on instances.

```
CATIA_occmap_tcproperties = bl_sequence_no, catiaOccurrenceName,
newOccurrenceNote
```

There is no default value.

For each occurrence note and BOMLine attribute, set **CATIA_occmap_xxx_catiaproperties**, where *xxx* corresponds to a value of **CATIA_occmap_tcproperties** preference to define:

- the instance property to map with this attribute. This value is case-sensitive.
- the qualifier to indicate the synchronization direction, i.e., **both**, **cad**, or **tc**. This value is not case-sensitive. When a qualifier is not defined, the value **both** is used.

Separate the property and qualifier values with a semi-colon (**;**). The value should have following syntax: `property;qualifier`

```
CATIA_occmap_xxx_catiaproperties = property;qualifier
```

Set the **_DescriptionInst** value in the preference to map the CATIA instance description, and set the **_InstanceName** value to map the CATIA instance name.

For instance:

```
CATIA_occmap_CustoNote1_catiaproperties=CATIAProp1;both

CATIA_occmap_CustoNote2_catiaproperties=CATIAProp2;cad

CATIA_occmap_BomLineAttr_catiaproperties=_DescriptionInst;tc
```

Integration occurrence notes defined in CATIA_filename_note, CATIA_occurrence_note, CATIA_parent_component_note, CATIA_uuid_note,

CATIA_occurrence_desc_note, CATIA_occurrence_oem_note and CATIA_occmapp_checksum_note preferences are protected data and can not be mapped.

5.7.1.8.6 CATIA behavior

Set **CATIA_force_update_instance_properties_for_occmapp** to indicate when CATIA instance properties should be updated for legacy data.

```
CATIA_force_update_instance_properties_for_occmapp = false
```

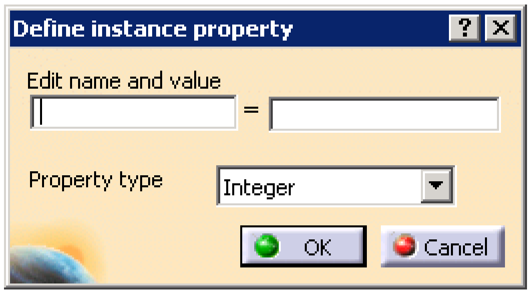
The default value is **false**.

Set **CATIA_create_catia_instance_properties_for_occmapp** to indicate when CATIA instance properties should be created when a mapping is defined on it.

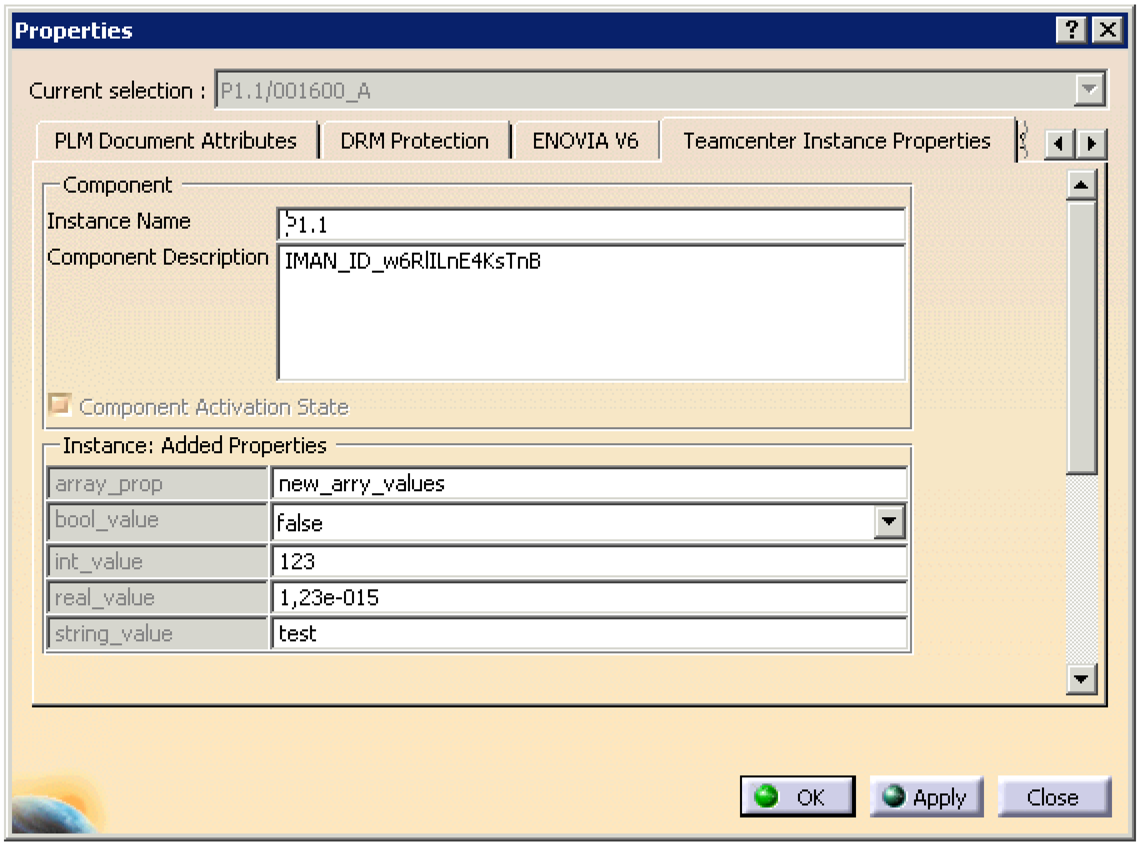
```
CATIA_create_catia_instance_properties_for_occmapp = false
```

The default value is **false**.

Instance properties can be created using **Cat2CreateInstanceProperty** command, delivered by the Integration. Launch the Cat2CreateInstanceProperty command in the CATIA power input or in the commands list available in **Tools >> Customize… >> Commands >> All commands** menu. Integer, Boolean or String property types can be created with this command.



Pick **More** button from Document Properties to display Instance properties in the Teamcenter Instance Properties tab.



Properties

Current selection : P1.1/001600_A

Component

Instance Name	P1.1
Component Description	IMAN_ID_w6RlILnE4KsTnB

Component Activation State

Instance: Added Properties

array_prop	new_arry_values
bool_value	false
int_value	123
real_value	1,23e-015
string_value	test

5.7.2 CGR Management

Overview

Configure CATIA

Configure Teamcenter

5.7.2.1 Overview

CGR (**.cgr**) is a lightweight visualization format adopted by CATIA to visually represent parts when accessed within the context of an assembly. CGR files consume less memory which is helpful with large assemblies.

When the **Work with the cache system** CATIA option is toggled on, CATIA creates CGR files for each CATPart in the local Cache directory, when an assembly is opened.

CGR and CATShape files can also be saved in Teamcenter during the **Save** process. This greatly improves the Load process performance and prevents CATIA from creating CGR files on the fly when the assembly is opened.

During the **Load** process, CATParts and CGR files are exported from Teamcenter, and CGR files are placed into the CATIA cache directory before the assembly is opened in CATIA, allowing faster load of the assembly.

Settings must be configured in order to create cache files.

The Integration's CGR Management functionality is available during the Save and Load processes.

CGRs can be stored locally or in cache.

5.7.2.2 Configuration

Activate the CGR Management functionality within the Integration by creating the following configurations in CATIA and in Teamcenter.

Both CATIA and Teamcenter must be configured.

5.7.2.3 Configure CATIA

Perform the following steps within CATIA to ensure CATIA cache is activated prior to using the CGR Management functionality.

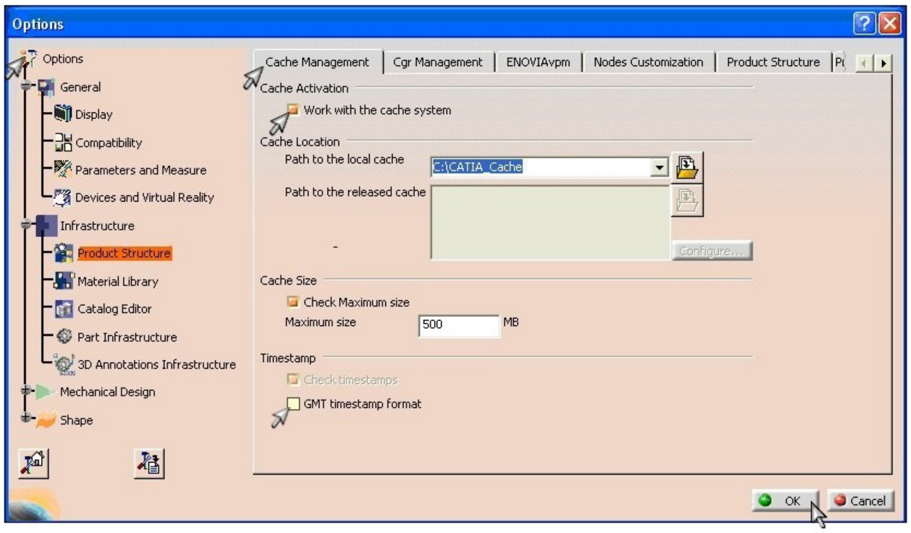
1. Pick **Options**.

2. Pick the **Cache Management** tab from the Options window.

3. Toggle the **Work with the cache system** option on.

4. It is recommended to enable the **GMT timestamp format** option to prevent failures during the Winter/Summer time change period. Using local time, rather than GMT, can cause time/date issues to occur.

5. Pick **OK** when finished.



Cache Activation
Work with the cache system
Cache Location
Path to the local cache: C:\CATIA_Cache
Path to the released cache
Cache Size
Check Maximum size
Maximum size: 500 MB
Timestamp
Check timestamps
GMT timestamp format

5.7.2.4 Configure Teamcenter

Ensure the following Teamcenter configuration/environment settings have been established prior to using the CGR Management functionality.

1. CATIA_cache_dataset_type

2. CATIA_cache_dataset_link

3. CATIA_cache_managed_on_load

4. CATIA_cache_managed_on_save

5. **CATIA_cache_path

6. **CATIA_CACHE_PATH_ENV *(environment setting)*

***Settings 5 and 6 define the export location.*

5.7.2.4.1 CATIA_cache_dataset_type

Name	Default Value	Purpose
CATIA_cache_dataset_type	CATCache	Dataset type used to store Cache files. This dataset type must contain two Named References. Values can be modified.

Refer to the following example to define a new dataset type called *CATCache2* containing a CGR Named Reference called *myCgr* and a CATShape Named Reference called *myCATshape*.

This newly created dataset type can be used to store Cache files by indicating `CATIA_cache_dataset_type` = `CATCache2`.

```
</TcDataset>
<TcDataset parentTypeName="Dataset" typeClassName="Dataset"
typeName="CATCache2">
<TcDSViewTool name="CatiaExport" />
<TcDSEditTool name="CatiaExport" />
<TcDatasetReference name="myCgr">
<TcDatasetReferenceInfo format="BINARY" template="*.cgr" />
</TcDatasetReference>
<TcDatasetReference name="myCatShape">
<TcDatasetReferenceInfo format="BINARY" template="*.CATShape" />
</TcDatasetReference>
</TcDataset>
```

5.7.2.4.2 CATIA_cache_dataset_link, CATIA_cache_managed_on_load, CATIA_cache_managed_on_save

Name	Default Value	Purpose
CATIA_cache_dataset_link	catia_cache_link	Link type used to link Cache dataset with CATPart/catia/jt/CATShape datasets

Name	Default Value	Purpose
CATIA_cache_managed_on_load	false	Do not manage Cache files during Load processes (false) **Teamcenter must be restarted to apply changes made to this preference.**
CATIA_cache_managed_on_save	false	Do not manage Cache files during Save processes (false) **Teamcenter must be restarted to apply changes made to this preference**.

5.7.2.4.3 Export Criteria

Refer to the criteria below to determine the location of exported CGR files based on defined settings.

The Integration determines the location to export CGR files based on the following criteria.

CGR files are exported to the **local cache** directory when any of the criteria is not met.

1. CGR files are exported to the path defined in the CATIA_CACHE_PATH_ENV environment variable when:
   - the path exists on your system and
   - the path is a released cache and
   - the user has write access.
2. If the CATIA_CACHE_PATH_ENV criteria is not met, CGR files are exported to the path defined in the CATIA_cache_path Teamcenter Preference when:
   - the path exists on your system and
   - the path is a released cache and
   - the user has write access.

3. If none of the above criteria is met, CGR files are exported to the local cache directory.

5.7.2.4.4 *Export Location*

Ensure the following settings have been established to correctly export CGR files.

Name	Default Value	Purpose
CATIA_cache_path	Local Cache path as defined within CATIA Options	Path where Cache files are exported to during Load processes.Refer to the export criteria for specific information. **The maximum length of the CATIA_cache_path is 240 characters.**
CATIA_CACHE_PATH_ENV *environment setting*	User defined	Path where Cache files are exported to during Load processes. Refer to the export criteria for specific information.

Use the CATIA_CACHE_PATH_ENV environment setting to define the path Cache files are exported to during Load processes.

```
CATIA_CACHE_PATH_ENV = C:\Released_Cache\Cache1
```

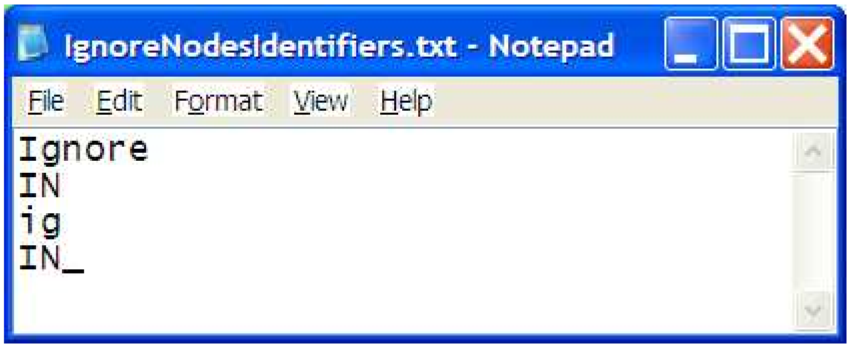
5.7.3 Configuring Ignore Nodes

A list of permissible Ignore Node prefixes must be manually defined in an *IgnoreNodesIdentifiers.txt* file.

The IgnoreNodesIdentifiers.txt file is not created during the installation of Teamcenter Integration for CATIA.

Perform the following steps to create the IgnoreNodesIdentifiers.txt file and place the file in its correct location.

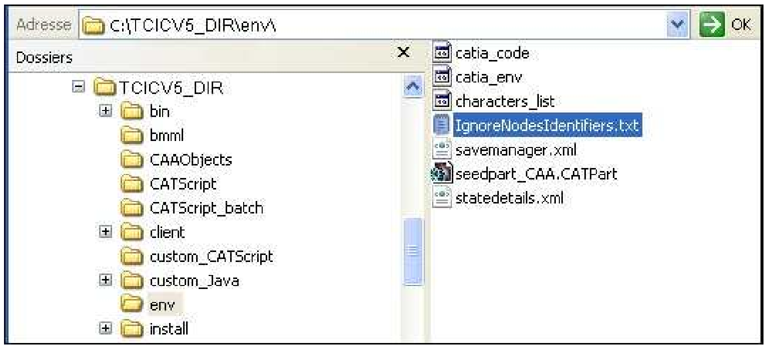
1. Create a new text file named *IgnoreNodesIdentifiers.txt* and enter one prefix per line. Prefixes are *case-sensitive*.


IgnoreNodesIdentifiers.txt - Notepad
File Edit Format View Help

```
Ignore
IN
ig
IN_
```

2. Save the IgnoreNodesIdentifiers.txt file in the following directory:

   - Windows:*%TCICV5_DIR%\env\*
   - UNIX: *$TCICV5_DIR/env/*



The Load for Compare prefix, comp_ , used in the CATIA_load_for_compare_prefix setting, must not be included in the list of Ignore Nodes prefixes.

5.7.4 Import Queries

Customize queries used during the Import process. Import forms are managed using a **GRM Link**. As a result, the import query returns the *form* type.

Previous to Teamcenter Integration for CATIA V5 (8.3.0), the import forms were managed as a Named Reference to the dataset. Beginning with Teamcenter Integration for CATIA V5 (8.3.0), import forms are now managed using a GRM Link, and as a result:

Import queries for CATProduct, CATPart, CATDrawing and CATDesignTables must be changed when installing the 8.3.0 version of the Integration.

Named Reference (catimportform) must be replaced by GRM (catia_import_form_link) for attribute mapping

An example of the new import query can be found in *TcIC_DIR\sample\ImportExport* and an example of the new attribute mapping field can be found in *TcIC_DIR\sample\ImportExport*.

5.7.4.1 Build an Import Query

Perform the following steps to build an import query.

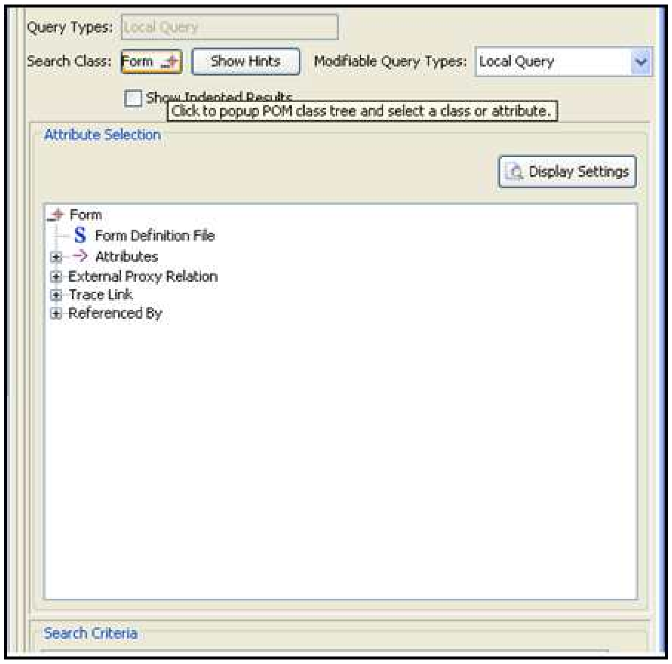
1. Log into Teamcenter with **Administrator** privileges.

2. Pick the **Admin** tab, then pick **Query Builder**.

3. Enter a name for your query in the Name field of the Query Builder form. Enter a description, if desired.

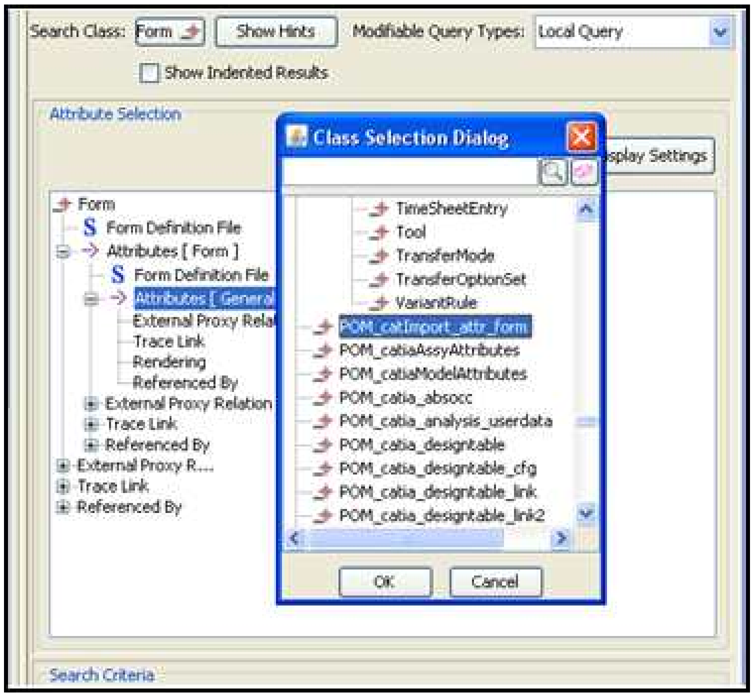
4. Pick the Search Class pull-down list.

5. Navigate or expand to **POM_application_Object >> WorkspaceObject >> Form** as the search class for your query.



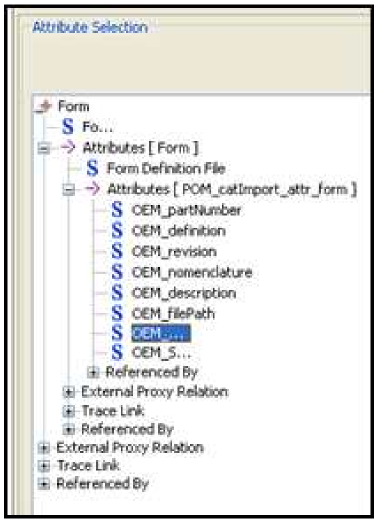
6. Double-click on **Attributes** within the Form node.

7. Navigate or expand to **POM_object >> POM_application_object >> POM_catimport_attr_form** to pick your import forms class definition.

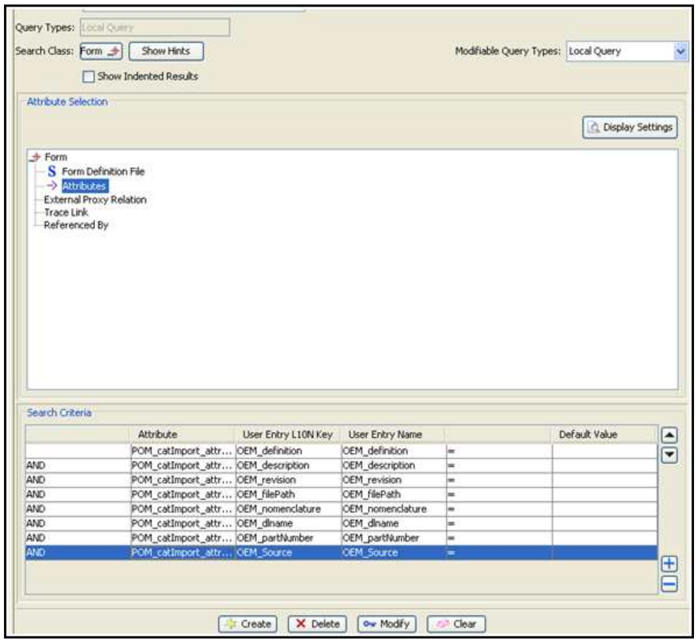


The attributes defined for your import forms (such as OEM_definition and OEM_revision) display within the Attributes Selection box

8. Double-click on each attribute that matches your requirements to add them to your Search Criteria.



9. Pick **Create** when finished.



Query Types: Local Query

Search Class: Form

Modifiable Query Types: Local Query

Show Indented Results

Attribute Selection

- Form
  - Form Definition File
  - Attributes
- External Proxy Relation
- Trace Link
- Referenced By

Search Criteria

	Attribute	User Entry L10N Key	User Entry Name		Default Value
	POM_catImport_attr…	OEM_definition	OEM_definition	=	
AND	POM_catImport_attr…	OEM_description	OEM_description	=	
AND	POM_catImport_attr…	OEM_revision	OEM_revision	=	
AND	POM_catImport_attr…	OEM_filePath	OEM_filePath	=	
AND	POM_catImport_attr…	OEM_nomenclature	OEM_nomenclature	=	
AND	POM_catImport_attr…	OEM_dlname	OEM_dlname	=	
AND	POM_catImport_attr…	OEM_partNumber	OEM_partNumber	=	
AND	POM_catImport_attr…	OEM_Source	OEM_Source	=	

Currently, these names should be exactly the same as the ones used for the import form attributes.

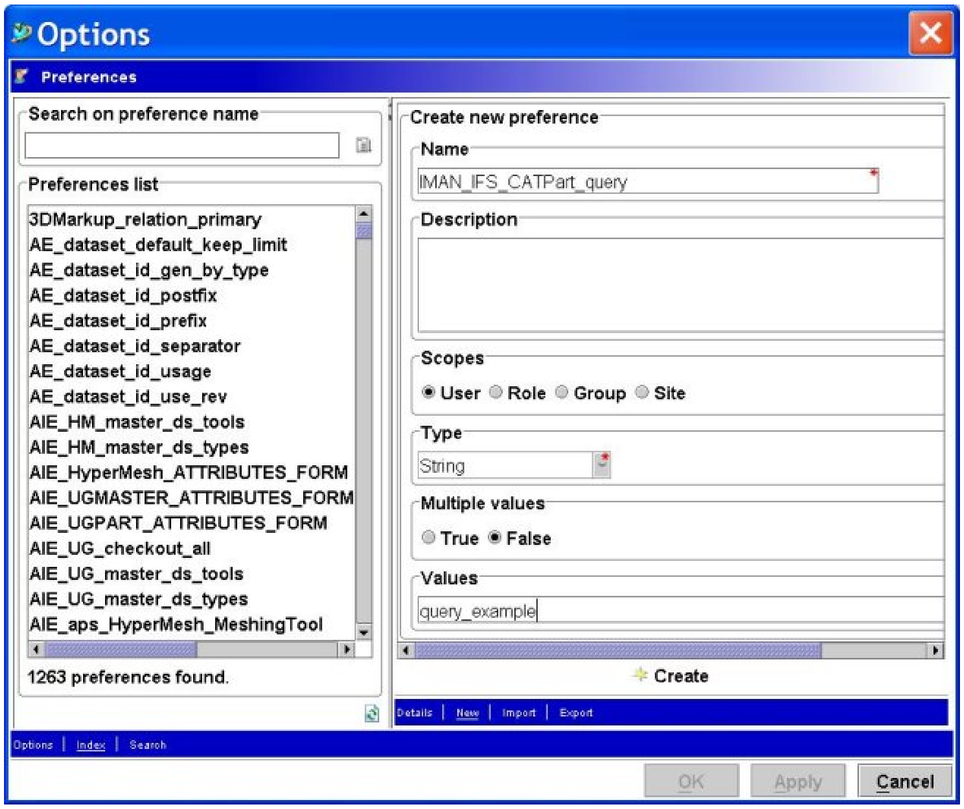
5.7.4.2 Create New Import Preferences

Teamcenter must be restarted after modifying preferences.

Modifying all Import preferences is not mandatory. However, query attributes names must match your `catImportForm` attributes names (case-sensitive), such as `OEM_filePath` , `OEM_definition`.

As a rule, mappings, form fields and query attributes names must be spelled exactly the same way (case-sensitive). Failure to do so may lead to inconsistent results.

Create New Preferences:



Name: IMAN_IFS_CATPart_query

Scopes: [✓ User] Role Group Site

Type: String

Multiple values: True [✓ False]

Values: query_example

- **CATIA_ifs_catpart_query** = name of query
- **CATIA_ifs_catproduct_query** = name of query
- **CATIA_ifs_catdrawing_query** = name of query
- **CATIA_ifs_designtable_query** = name of query
- **CATIA_ifs_foreignfile_query** = name of query

When the individual queries listed below are created, it is not necessary to create the preferences listed above (CATIA_ifs_).

- ifs_catdrawing_query
- ifs_catproduct_query
- ifs_catpart_query
- ifs_designtable_query

- ifs_foreignfile_query

The Integration executes your queries using the following fields of the Import form:

- OEM_partNumber
- OEM_definition
- OEM_revision
- OEM_nomenclature
- OEM_description
- OEM_filePath
- OEM_dlname
- OEM_Source

When different names and/or different fields for your queries are required, modify Teamcenter Preferences by adding the following and the corresponding catImport form fields used:

Multiple values are permitted for these preferences.

```
CATIA_ifs_catpart_query = the name of the query
```

The default value is **IFS_CATPart_query**.

```
CATIA_ifs_catproduct_query = the name of the query
```

The default value is **IFS_CATProduct_query**.

```
CATIA_ifs_catdrawing_query = the name of the query
```

The default value is **IFS_CATDrawing_query**.

```
CATIA_ifs_designtable_query = the name of the query
```

The default value is **IFS_CATDesignTable_query**.

```
CATIA_ifs_foreignfile_query = the name of query
```

The default value is **IFS_CATForeignFile_query**.

```
CATIA_ifs_catpart_query_attr =
attr1
attr2
...
```

The default values are **OEM_partNumber**, **OEM_definitions**, **OEM_revision**, **OEM_nomenclature**, **OEM_description**, **OEM_filePath**, **OEM_dlname**.

```
CATIA_ifs_catproduct_query_attr =
attr1
attr2
...
```

The default values are **OEM_partNumber**, **OEM_definition**, **OEM_revision**, **OEM_nomenclature**, **OEM_description**, **OEM_filePath**, **OEM_dlname**.

```
CATIA_ifs_catdrawing_query_attr =
attr1
attr2
...
```

The default values are **OEM_filePath**, **OEM_dlname**.

```
CATIA_ifs_designtable_query_attr=
attr1
attr2
...
```

The default values are **OEM_filePath**, **OEM_dlname**.

```
CATIA_ifs_foreignfile_query_attr=
attr1
attr2
...
```

The default values are **OEM_filePath**, **OEM_dlname**.

5.7.4.3 Item Revision Browser Queries

Item revisions queries must be defined for the Item Revision Browser, which displays during the Save process. Queries can be imported from an existing .xml file, as detailed below, or they can be manually created.

Perform the following steps to import the queries from an existing .xml file.

1. Log into Teamcenter with **Administrator** privileges.
2. Pick the **Admin** tab, then pick **Query Builder**.
3. Pick **Import**.
4. **Browse** on the Read Query Definition window to locate the ItemRevBrowser_query.xml file in the *TcIC_DIR\sample\ItemRevBrowser* folder.
5. Pick the file, then pick **Import**. Ensure it is the correct file, then pick **OK** on the Import window to continue.
6. Pick **Create** on the Query Builder.

5.7.4.3.1 Manually Build the Item Revision Browser Query

Supplier Import Queries must adhere to the following naming convention:

- ItemRevBrowser_query

5.7.4.4 Supplier Import Queries

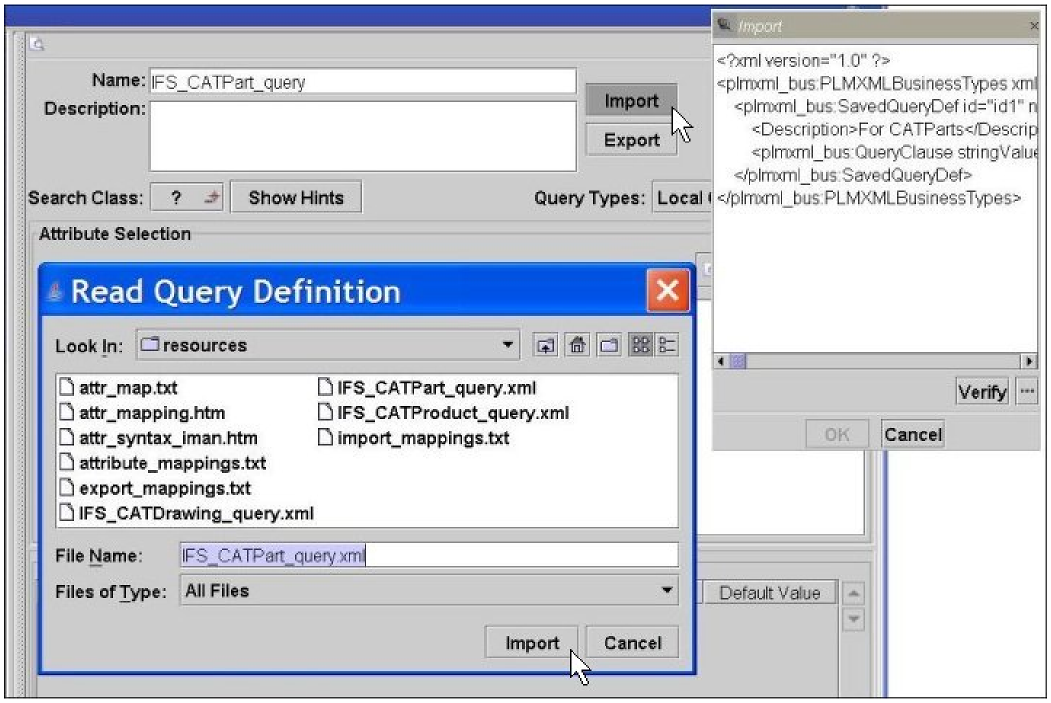
Four import queries must be defined for supplier import functions. The queries can be imported from an existing .xml file, as detailed below, or they can be manually created.

Perform the following steps to import the queries.

1. Log into Teamcenter with **Administrator** privileges.
2. Pick the **Admin** tab, then pick **Query Builder**.
3. Pick **Import**.
4. **Browse** on the Read Query Definition window to locate the four .xml files within the *TcIC_DIR\sample\ImportExport* folder.
    - *IFS_CATDrawing_query.xml*
    - *IFS_CATPart_query.xml*

- *IFS_CATProduct_query.xml*
- *IFS_CATDesignTable_query.xml*
- *IFS_CATForeignFile_query.xml*

5. Pick one of the files, then pick **Import**. Ensure it is the correct file, then pick **OK** on the Import window to continue.
6. Pick **Create** on the Query Builder.
7. Repeat the previous steps to import the remaining two files.



Name: IFS_CATPart_query
Query Types: Local
Look In: resources

attr_map.txt
attr_mapping.htm
attr_syntax_iman.htm
attribute_mappings.txt
export_mappings.txt
IFS_CATDrawing_query.xml
IFS_CATPart_query.xml
IFS_CATProduct_query.xml
import_mappings.txt

File Name: IFS_CATPart_query.xml
Files of Type: All Files

```
<?xml version="1.0" ?>
<plmxml_bus:PLMXMLBusinessTypes xml
  <plmxml_bus:SavedQueryDef id="id1" n
    <Description>For CATParts</Descrip
    <plmxml_bus:QueryClause stringValue
  </plmxml_bus:SavedQueryDef>
</plmxml_bus:PLMXMLBusinessTypes>
```

5.7.4.4.1 Manually Build the Supplier Import Queries

Supplier Import Queries must adhere to the following naming conventions:

- IFS_CATDrawing_query
- IFS_CATPart_query
- IFS_CATProduct_query

- IFS_CATDesignTable_query
- IFS_CATForeignFile_query

5.7.5 Developing the Integration Schema

This section explains how to manually configure the Teamcenter Database schema to customize the Teamcenter Integration for CATIA V5 software on your client workstations.

Refer to Teamcenter Database section for information on applying the default configuration of the Teamcenter database for use with Teamcenter Integration for CATIA V5.

Teamcenter Integration for CATIA V5 is case-sensitive.

5.7.5.1 The Integration Schema

The following attributes are *automatically completed* by Teamcenter Integration for CATIA when the **TC_Allow_Longer_ID_Name** Teamcenter Preference is set to **false** (default).

Attributes names	Type	Length	Notes
CATIA OCCURRENCE NAME	CHAR	---	Contains the CATIA V5 occurrence name.
DATASET	CHAR	Up to	---

Attributes names	Type	Length	Notes
DESCRIPTION		72	
DATASET NAME	CHAR	32	Same as item ID.
DATASET VERSION	CHAR	---	Automatically incremented.
ITEM ID	CHAR	32	Given by Teamcenter Integration for CATIA.
ITEM NAME	CHAR	32	The first 32 characters of the CATIA V5 product, part or drawing.
ITEM REVISION	CHAR	---	---
PSE SEQUENCE NUMBER	CHAR	---	---

The following attributes are *automatically completed* by Teamcenter Integration for CATIA when the **TC_Allow_Longer_ID_Name** Teamcenter Preference is set to **true**:

Attributes names	Type	Length	Notes
CATIA OCCURRENCE NAME	CHAR	---	Contains the CATIA V5 occurrence name.
DATASET DESCRIPTION	CHAR	Up to 72	---

Attributes names	Type	Length	Notes
DATASET NAME	CHAR	128	Same as item ID.
DATASET VERSION	CHAR	---	Automatically incremented.
ITEM ID	CHAR	128	Given by Teamcenter Integration for CATIA.
ITEM NAME	CHAR	128	The first 32 characters of the CATIA V5 product, part or drawing.
ITEM REVISION	CHAR	---	---
PSE SEQUENCE NUMBER	CHAR	---	---

5.7.5.2 How Teamcenter Integration for CATIA Manages CATIA Components

The following can be **stored in Teamcenter** for each CATIA product, part or drawing:

- An item and an item revision.
- A dataset and CATProduct, CATPart or CATDrawing, and a dataset version.
- A specific link between an item revision and a dataset.
- The BOM view attached to the created item for an assembly.
- A BOM view revision attached to the created item revision.
- A **catia_model_attribute** type attached to the item revision for a part.
- A **catia_assembly_attribute** type for a product.

- A **catiaOccurrenceName** note containing the CATIA occurrence name of the part or the product.

The **relative positions** of parts or products are stored within the occurrence of the item revision of the BOM during a CATIA session.

Several datasets can be linked to one item. However, a referenced CATPart and a referenced CATProduct cannot be linked to the **same** item revision. Use standard functionalities of Teamcenter to link a referenced CATpart and a referenced CATProduct to the same item revision.

5.7.5.3 Create the CatiaExport Tool

Use the **Business Modeler IDE** (BMIDE) to create the template and configure the database schema.

The template contains the following:

- CatiaExport Tool

5.7.5.4 Seed Documents

Item ID names are defined in the following Teamcenter Preferences:

- CATIA_seed_part_id
- CATIA_seed_product_id
- CATIA_seed_drawing_id
- CATIA_seed_analysis_id
- CATIA_seed_catshape_id
- CATIA_seed_process_id

5.7.5.4.1 Create a Seed Document

There are two methods to create a seed document.

1. In Teamcenter
2. During the Integration Save process

Any dataset of a corresponding seed document may be loaded when datasets containing seed documents exist and are loaded without a Named Reference.

A dataset cannot be loaded without a Named Reference when seed documents are not defined and when the CATIA Manual input Part Number option not is selected from the Options dialog. In this case, Seed documents must be defined.

Due to a CATIA limitation, it is not possible to have Dynamic Seeds for a CATDrawing and activate Auxiliary Files, in combination.

5.7.5.4.2 Create a Seed Document During the Integration's Save Process

Perform the following steps with Administrator privileges to create a seed document during the Integration's Save process.

1. In Teamcenter, create a new item. Ensure the item ID matches the seed preference value, for example, ensure the item ID matches the value of CATIA_seed_part_id.
2. Add a new dataset to the item just created.
3. In CATIA, create or open a document to use as the seed document.
4. Pick the **Save to Teamcenter** command in CATIA.
5. Ensure the Dataset Name field in the Save Manager relates to the dataset just added, then continue to Save process.

5.7.5.4.3 Create a Seed Document in Teamcenter

A seed item can contain several datasets of the same type. An Integration seed panel displays when several dataset are found during a load process, to select the correct seed document to load.

Perform the following steps with Administrator privileges to create a seed document in Teamcenter.

1. Create the item(s) or item revision(s) in Teamcenter. Ensure it has the same ID as the value defined in the preference for the seed document type.

2. Create the CATpart, CATProduct, CATDrawing, CATAnalysis, CATShape or CATProcess as needed.

3. Add a Named Reference to the dataset. Ensure a valid Named Reference is added based on the seed type.

5.7.5.4.4 Multiple Seed Documents

Multiple seed documents can be attached to each item type; CATPart, CATProduct, CATDrawing and CATShape. When multiple seed documents are available for a given type, pick the appropriate seed document. When loading from the PSE, pick the same seed document for individual parts/products or for all parts/products.

The seed document must be created and attached to the dataset first, prior to creating multiple seeds.

Perform the following steps to create multiple seed documents for a CATPart.

1. Ensure the first seed document has been created.

2. In Teamcenter, add a new dataset with the same dataset type as the existing dataset under the IMCATIAPART/A item revision, i.e., the current seedpart.

3. Add any Named Reference to the dataset just created.

4. Load the dataset in CATIA using the Integration.

5. In CATIA, remove or modify the geometry as needed.

6. Pick a **Save** process from Teamcenter.

7. Ensure the Dataset Name field in the Save Manager relates to the dataset added under the IMCATIAPART/A, then continue the Save process.

5.7.5.4.5 *Integration Toolbar*

Configure Environment Messages
Configure Icons

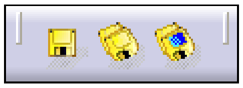
See Also:

Integration Toolbar: Icon Descriptions and File Names

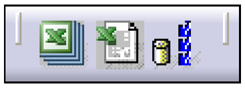
The Integration toolbar comprises different toolbars.

Save
Import
Load
Add CATShape
Utility
Replace Load as cgr

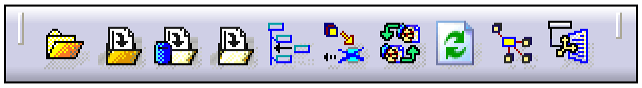
5.7.5.5 Save



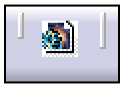
5.7.5.6 Import



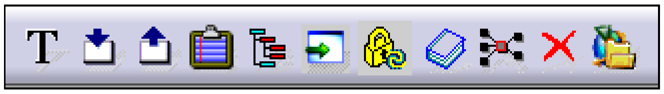
5.7.5.7 Load



5.7.5.8 Add CATShape



5.7.5.9 Utility



5.7.5.10 Replace Load as cgr

5.7.5.11 Configure Environment Messages

Perform the following steps to change an icon's tooltip and command displayed in the status bar.

1. Edit the *TeamcenterHeader.CATNls* file located in the *%TCICV5_DIR%\bin\CAA\V5R<CATIA version>\toolbar\intel_a\resources\msgcatalog* directory.

2. The following lines display for each icon/command defined in the toolbar. In the example below, **Save** indicates the icon/command associated to these lines:

```
TeamcenterHeader.Save.Title = "Cmd Teamcenter"
 TeamcenterHeader.Save.ShortHelp = ""
 TeamcenterHeader.Save.Help = ""
 TeamcenterHeader.Save.LongHelp = ""
```

5.7.5.12 Configure Icons

Perform the following steps to associate an icon to a specific menu command.

1. Save the new icon in the *%TCICV5_DIR%\bin\CAA\V5R<your CATIA version>\toolbar\intel_a\resources\graphic\icons\normal* directory.

2. Open the **TeamcenterHeader.CATRsc** file located in the *%TCICV5_DIR%\bin\CAA\V5R<your CATIA version>\toolbar\intel_a\resources\msgcatalog* directory.

3. Modify the TeamcenterHeader.CATRsc file by replacing the current name of the icon with the new name.

For example, **Load**, in the line below, refers to the Load command and **I_01001UGS_Load** indicates the icon name associated to the Load command. Change the I_01001UGS_Load icon name to the new icon to associate it to the Load command.

```
TeamcenterHeader.Load.Icon.Normal = "I_01001UGS_Load"
```

5.7.6 Configuring Multiple CATIA Environments

Customize CATIA Running Scripts
Launch Different CATIA Versions
View the Log Files
Preferences Related to Multiple CATIA Environments

See Also:

Add/Remove Save Manager Columns

Multiple CATIA environments are supported and can be customized for specific environments in order to:

- prevent saving an assembly back to Teamcenter in a different CATIA environment than when previously saved
- prevent delivering data to CATIA that will fail to load due to different CATIA environments
- launch CATIA using the correct version and environment during batch processes.

5.7.6.1 Customize CATIA Running Scripts

***View the Log File* before proceeding.**

Perform the following modifications to the CATIA running script to control multiple CATIA environments.

5.7.6.1.1 The LaunchCatia.bat script for Windows

Use the `LaunchCatia.bat` script in Windows to permit/prevent CATIA from being launched in a batch mode and to run an interactive CATIA session (use the last launched CATIA version). This script can also be modified to prevent CATIA from starting from Teamcenter.

Permit/Prevent Launch in Batch Mode and Run an Interactive Session

Perform the following modifications to the Windows script, as required.

Remove the **%6** from the line shown below and replace it with **-batch** to launch CATIA in Batch mode.

When %6 is removed from the line below, CATIA launches in Interactive mode.

```
%3\CATSTART.exe -run "CNEXT.exe" -env %4 -direnv %5 -object " %6 -
macro \"%MACRO_PATH%\" "
echo LaunchMacro.bat ended >> %LOG_FILE%
exit 0

%3\CATSTART.exe -run "CNEXT.exe" -env %4 -direnv %5 -object " -
batch  -macro \"%MACRO_PATH%\" "
echo LaunchMacro.bat ended >> %LOG_FILE%
exit 0
```

Modify the path name of the script by modifying the `LOG_File` variable as shown below.

```
@echo off
REM
*********************************************************************
*********
REM %1 NewSession (run a new CATIA), ExistingSession (only for
windows, use an existing CATIA session)
REM %2 name of the CATScript's macro
REM %3 CATIA pathname (needed if %1==NewSession)
REM %4 environnement (needed if %1==NewSession)
REM %5 environment pathname (needed if %1==NewSession)
REM %6 batch mode (needed if %1==NewSession)
REM
*********************************************************************
*********
REM Init
set LOG_FILE="%WORK_DIR%\tmp\LaunchMacro.bat.out"

REM Remove old output file
IF EXIST %LOG_FILE% del %LOG_FILE%
```

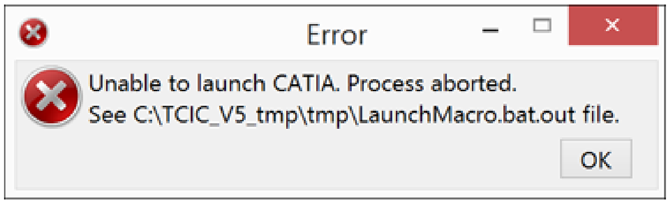
Prevent CATIA from Starting from Teamcenter

Modify `LaunchCATIA.bat` script as follows to prevent CATIA from starting from Teamcenter.

```
:NEWSESSION
echo New session >> %LOG_FILE%
if X%2==X goto PARAM2_ERROR
if X%3==X goto PARAM3_ERROR
if X%4==X goto PARAM4_ERROR
if X%5==X goto PARAM5_ERROR
```

```
REM %3\CATSTART.exe -run "CNEXT.exe" -env %4 -direnv 5% -object "
%6 -macro \"%MACRO_PATH%\" "
goto END
REM echo LaunchMacro.bat ended >> %LOG_FILE%
REM exit 0
```

An error message will display during a load from Teamcenter.


Error

Unable to launch CATIA. Process aborted.
See C:\TCIC_V5_tmp\tmp\LaunchMacro.bat.out file.

5.7.6.2 Launch Different CATIA Versions

Different CATIA versions can be launched within the Integration, but must be successfully defined by copying and editing configuration files and shortcuts.

Refer to your Administrator and to the Define Different CATIA Versions section to permit different CATIA versions to be launched within the Integration.

Launch a different version of CATIA using a shortcut.

Launching CATIA from Teamcenter defaults to the last version of CATIA installed using the Integration.

5.7.6.3 View the Log File

Before performing modifications to the CATIA Running Scripts, view the appropriate log file listed below to determine parameters used to run CATIA, the name of the CATScript to launch, etc.

Log File Name	File Location
`LaunchMacro.bat.out`	**Windows**: `%WORK_DIR%\tmp`

Log File Name	File Location
`LaunchMacro.sh.out`	**UNIX**: `$WORK_DIR/tmp`

5.7.6.4 Preferences Related to Multiple CATIA Environments

Teamcenter must be restarted after modifying preferences.

5.7.6.4.1 CATIA Environment Compatibility

Customize the **CATIA_CE_Alias_xxx** setting, where *xxx* is the environment found in the catia_doc_attributes form, to indicate the CATIA environment is compatibility

```
CATIA_CE_Alias_Ugs_ce = Ugs_ce_2
```

The example above indicates that CATIA is running with the **Ugs_ce_2** environment and as a result:

If...	Then
an item containing a form is Saved into Teamcenter **and** the form indicates `OEM_CATIA_Env = Ugs_ce_2`	The item in CATIA and the Teamcenter environment are compatible.

Both CEs (Ugs_ce and Ugs_ce_2) must be defined within the CATIA_CE_EnvList, using the same spelling.

5.7.6.4.2 Check the Environment During the Load/Save Process

Use **CATIA_check_env** to control multiple CATIA environments by checking the environment during the load/save processes.

```
CATIA_check_env = NONE
```

The default value is **NONE**.

Value	Result
NONE	Default value. Multiple CATIA environments and CATIA revisions are not managed. • Teamcenter does not save CATIA revisions • Loading of data to CATIA occurs without CATIA revision checks (leaving the process to native CATIA) • Data loaded may be saved to Teamcenter regardless of last saved version. • New data may always be saved using the current revision and environment.
REV	The CATIA version is only managed. The CATIA version and environment is written to the form during a **Save** operation. The CATIA version is read from the form during a **Load** operation. • Multiple CATIA revisions are managed for the Load process • Multiple CATIA environments are not managed for the Save process • Loading of data to CATIA is prevented when the active CATIA revision is **lower** than the saved revision • Data loaded may be saved to Teamcenter regardless of last saved revision. • New data may always be saved using the current revision and environment.

Value	Result
ENV	The CATIA version and environment are managed. The CATIA version and environment are written to the form during a **Save** operation. The CATIA version is read from the form during a **Load** operation. If a CATIA session does not exist, the environment written in the form of the root is used to launch CATIA with the correct environment. • Multiple CATIA revisions are managed for the Load process • Multiple CATIA environments are managed for the Save process • The current CATIA environment must be declared • Teamcenter records the CATIA revision and environment • Loaded data to CATIA is prevented when the active CATIA revision is **lower** than the saved revision • Data loaded may be saved to Teamcenter provided the current CATIA environment or defined Alias matches the data being saved • New data may always be saved using the current revision and environment. ***The CATIA_check_env fails when its value is set to ENV and an alias has not been defined. It will not fail if it is set to REV and an alias has not been defined.***

If CATIA is **not launched**, then the following process determines the correct CATIA environment.

- CATIA_check_env preference
- The catia_env file
- The root item of the assembly being loaded.

Value	Result
NONE	Default value. Multiple CATIA environments are not managed. The environment used is the default environment as defined within the `<ENV></ENV>` tags.

Value	Result
REV	The Integration searches the root item of the assembly if the item to be loaded was selected from the PSE, otherwise the root item is the selected item from My Teamcenter. The Integration reads the `OEM_CATIA_Version` field value of the root item and searches the first environment corresponding to its value. If the search fails, the **default** environment is used.
ENV	The Integration searches the root item of the assembly if the item to be loaded was selected from the PSE, otherwise the root item is the selected item from My Teamcenter. The Integration reads the `OEM_CATIA_Env` field value of the root item and searches the first environment corresponding to its value. If the search fails, the **default** environment is used.

5.7.7.3 Define the Environment and the Nickname Written in Teamcenter

Customize the **CATIA_CE_EnvList** setting.

Multiple values are permitted.

The CATIA_CE_EnvList setting must contain the following values.

1. The name of the CATIA text file that contains the CATIA environment definition (i.e., Teamcenter_manager_for_CATIAV5.txt located within the *All Users\Application Data\DassaultSystemes\CATEnv* directory). For assistance in locating this file, please see your CATIA Administrator.

2. The nickname that will be written in Teamcenter.

Its value **must** include both the CATIA_env_name and alias, for example:

```
CATIA_CE_EnvList =
CATIA_env_name_1
Alias_1
```

The CATIA environment and a CE associated to the CATIA environment must be defined within this setting.

```
CATIA_CE_EnvList =
Teamcenter_manager_for_CATIA_V5
Ugs_ce
```

Multiple environments may also be defined within this setting. Ensure the setting contains one environment name followed by a nickname.

```
CATIA_CE_EnvList =
Teamcenter_manager_for_CATIA_V5
Ugs_ce
CATIA_R16_CAA
Ugs_ce_2
```

5.7.7 Configuring the Theorem/Teamcenter Translator

5.7.7.1 The catia2jt script for the Theorem Translator

Perform the following steps to configure the **catia2jt** script for the Theorem translator.

1. Locate the **catia2jt.bat** or **catia2jt** file in the following Integration install directory: *%TCICV5_DIR%\bin\WIN32* or *$TCICV5_DIR/bin/{Os}*.

2. Ensure the correct translator path for *TS_INST=C:\Translator_Path* is included in catia2jt.bat or catia2jt.

3. Define **CATIAV5_ENV** and **CATIAV5_DIRENV**. For example, *CATIAV5_ENV = Teamcenter_Integration_for_CATIAV5* and *CATIAV5_DIRENV = C:\apps\catia\R19\CATEnv*.

4. Ensure the correct V5_EXE/V4_EXE line is activated in the catia2jt.bat or catia2jt file (Siemens or Theorem).

5.7.7.2 Call catiav5tojtdirect Service in Workflow

The **TSTK-CreateTranslationRequest** handler must be configured to call the **catiav5tojtdirect** service in workflow.

A handler for each DatasetType must be defined separately in workflow in order to create a JT for each CATIA dataset.

TSTK-CreateTranslationRequest handler syntax:

```
ProviderName=SIEMENS
ServiceName=Catiav5tojtdirect
Priority=1
DatasetTypeName=CATPart
```

5.7.7.3 Preferences Related to Dispatcher Services

Teamcenter must be restarted after modifying preferences.

Use **DISPATCHER_rqst_on_specific_dataset_revision** to use a specific revision (**true**) or the latest revision (**false**, default) of a dataset during Dispatcher Services.

```
DISPATCHER_rqst_on_specific_dataset_revision = false
```

The default value is **false**.

6.0 Customizing Teamcenter Integration for CATIA V5

6.1 BOMView Type

Teamcenter must be restarted after modifying preferences.

Customize the BOM View Type by setting **CATIA_Bom_name** within Teamcenter Preferences.

```
CATIA_Bom_name= newBomView
```

The default value is **catia**.

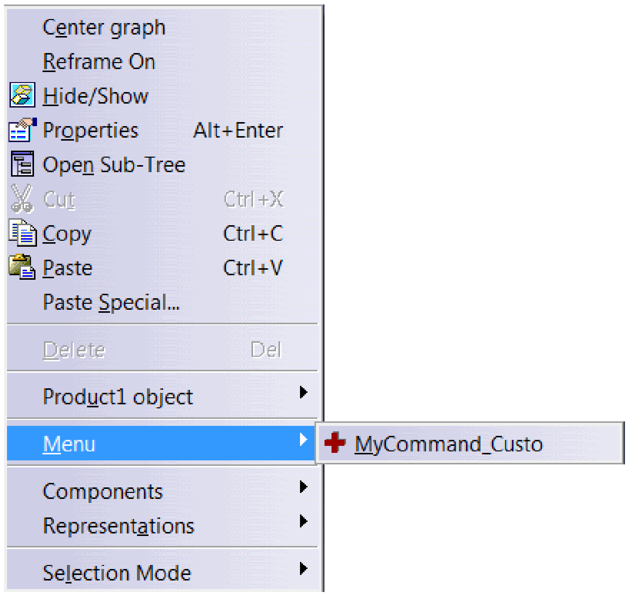
6.2 Create and Customize a Contextual Menu or Toolbar

[Create a Contextual Menu](#)
[Customize a Contextual Menu](#)
[Create a New Toolbar](#)

6.2.1 Create a Contextual Menu

Refer to this section as a guide to create a new contextual menu and insert commands.

The Teamcenter menu in the CATIA toolbar can be customized like all standard contextual menus in CATIA.



Implementing the **CATIContextualMenu** interface involves two main steps:

1. Create the Contextual Menu Description Class
2. Create the Contextual Menu

6.2.1.1 Create the Contextual Menu Description Class

1. Create the CTXMenuProductAddin.h file. The implementation class is derived from the CATICtxMenuProvider adapter class.

```
#include "TIE_CATICtxMenuProvider.h"
class ExportedByCADPolToolbarContextualMenuAddin
CTXMenuProductAddin :
public CATBaseUnknown
{
   CATDeclareClass;

public :

   /**
    * Class constructor
    */
   CTXMenuProductAddin ();

   /**
    * Class destructor
    */
   virtual ~CTXMenuProductAddin ();

   /**
    * Contextual Menu creation
    * @return A pointer to the contextual menu created
    */
   HRESULT GetContextualMenu(CATBaseUnknown*  i_object,
                             CATCmdContainer* o_contextual_menu);

private :

   /**
    * Set the titles to the contextual menu
    */
   HRESULT SetContextualMenuTitle();

   /**
    * Copy constructor, must not be implemented <br>
    * Set as private to prevent from compiler automatic creation as
public
    * @return A pointer to the contextual menu
    */
   CTXMenuProductAddin (CTXMenuProductAddin&);
   /**
    * Assigment operator, must not be implemented <br>
    * Set as private to prevent from compiler automatic creation as
public
    * @return A pointer to the contextual menu
    */
   CTXMenuProductAddin& operator=(CTXMenuProductAddin&);
};
```

2. Create the CTXMenuProductAddin.cpp file.

```
/*-------------------------------------------------------------------
----------*/
/*Creates the CADPolToolbarHeader command header class*/
MacDeclareHeader(CADPolToolbarHeader);
/*-------------------------------------------------------------------
----------*/
/*Declares the implementation of CTXMenuAddin class*/
CATImplementClass(CTXMenuProductAddin,
                  Implementation,
                  CATBaseUnknown,
                  CATnull);
/*-------------------------------------------------------------------
----------*/
/*Declares the interfaces implemented by CTXMenuProductAddin
class*/
#include "TIE_CATICtxMenuProvider.h"
TIE_CATICtxMenuProvider(CTXMenuProductAddin);...
```

The `CTXMenuProductAddin` class states that it implements the `CATICtxMenuProvider` interface because of the `TIE_ CATICtxMenuProvider` macro.
The `CATImplementClass` macro declares that the `CTXMenuProductAddin` class is an implementation, due to the implementation keyword. The third argument must always be set as `CATBaseUnknown` or `CATNull` for any kind of extension.

The constructor contains the main code and the destructor is empty.

```
...
CTXMenuProductAddin::CTXMenuProductAddin () : CATBaseUnknown()

{
   Creating the Contextual Menu
}

CTXMenuProductAddin::~CTXMenuProductAddin () {
}
```

3. Update the interface dictionary, for example, the CADPolToolbar.dico file. The directory's pathname is concatenated at run time in the CATDictionaryPath environment variable. It declares that the CTXMenuProductAddinCusto component implements the CATICtxMenuProvider interface, whose code is located in the libCADPolToolbarContextualMenuAddin shared library or DLL.

```
CTXMenuProductAddin      CATICtxMenuProvider       libCADPolToolbar
ContextualMenuAddin
```

6.2.1.2 Create the Contextual Menu

In this use case, the contextual menu associated with the UIActive object is retrieved first due to the GetContextualMenu method of the adapter class. This menu, p_contextual_menu, is completed with one command (MyCommand_Custo). A separator is also added.

```
  ...
/*-------------------------------------------------------------------
----------*/
/*Creates the add-in and arrange the commands of the Contextual
Menu*/
HRESULT
CTXMenuProductAddin::GetContextualMenu(CATBaseUnknown*  i_object,
                                              CATCmdContainer*
o_contextual_menu)
{
   CATUnicodeString contextual_menu_title;
   /*contextual menu pointer*/
   NewAccess(CATCmdContainer, p_contextual_menu, TcCtxMenuAddin);


   /*add-in and command for Load Merge Selected Level*/
   NewAccess        (CATCmdStarter, p_command1, command1);
   SetAccessCommand (p_command1, "MyCommand_Custo");
   AddAccessChild   (p_contextual_menu, p_command1);

   /*separator*/
   NewAccess        (CATCmdSeparator, p_separator1, separator1);
   AddAccessChild   (o_contextual_menu, p_separator1);

   /*set the contextual menu title*/
   contextual_menu_title =
CATMsgCatalog::BuildMessage("CADPolToolbarHeader",
              "CADPolToolbarHeader.CtxMenu");
   p_contextual_menu->SetTitle(contextual_menu_title);
   SetContextualMenuTitle();

   /*set the contextual menu*/
   SetAccessNext(p_separator1, p_contextual_menu);
```

```
    return S_OK;
}    ...
```

The p_contextual menu, is completed because of the macros contained in the CATCreateWorkshop file:

NewAccess

- A command starter is created as a CATCmdStarter instance using the NewAccess macro. The p_command1 variable handles a pointer to the instance, and command1 is its identifier.

  A separator access is created as a CATCmdSeparator instance using the NewAccess macro. The pSeparator1 variable handles a pointer to the instance and separator1 is its identifier.

SetAccessCommand

- A command header is associated with a command starter using the SetAccessCommand macro. The second parameter of the macro is the command header identifier defined as the first parameter of the command header constructor. For example, MyCommand_Custo.

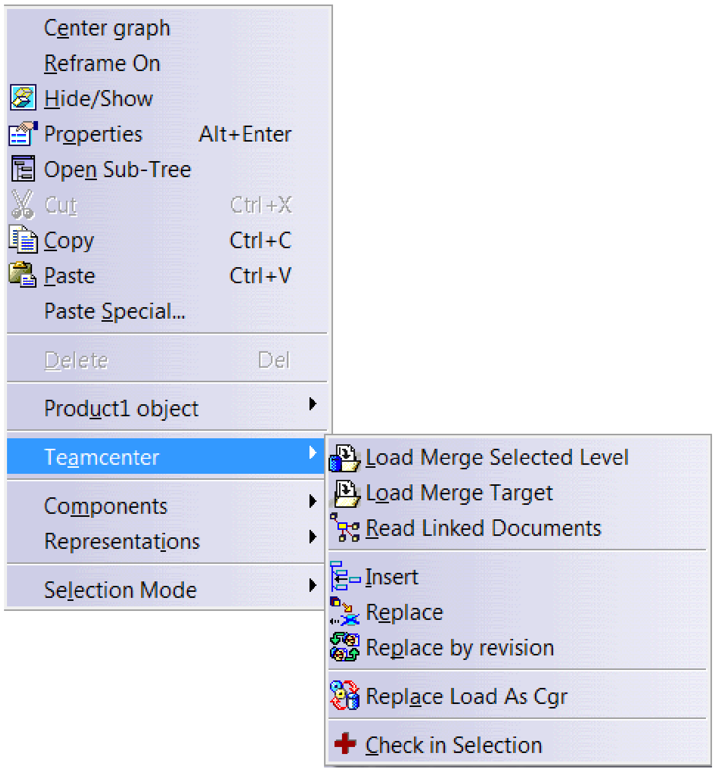
AddAccessChild/SetAccessNext

- The AddAccessChild macro enables linking the p_command1 access to the last access of _p_contextual_menu. The SetAccessNext macro enables chaining the other accesses to the p_command1 access.

6.2.2 Customize a Teamcenter Contextual Menu

Refer to this section as a guide to insert commands into the Teamcenter contextual menu of an object. The Teamcenter contextual menu provides quick shortcuts to common actions which can be performed on the selection.

The Teamcenter menu in CATIA can be customized like all standard contextual menus in CATIA. Modify LBTcMenuAddin.cpp file of the %TCICV5_DIR%/…/ CV5CAAToolbarTcMenuAddin.m module to customize your own implementation



.

Use the `GetContextualMenu` method to insert additional commands and separator.

1. Modify CTXMenuProductAddin.cpp file of the CV5CAAToolbarContextualMenuAddin.m module, provided in the CD-ROM extraction kit, (*CV5CAAToolbarContextualMenuAddin.m\src\CTXMenuProductAddin.cpp*)to add your own command “MyCommand_Custo” after the Highlight in Teamcenter command and a separator.

```
   ...
/*------------------------------------------------------------------
----------*/
/*Creates the add-in and arrange the commands of the Contextual
Menu*/
HRESULT
CTXMenuProductAddin::GetContextualMenu(CATBaseUnknown*  i_object,
                                              CATCmdContainer*
o_contextual_menu)
{
   CATUnicodeString contextual_menu_title;
   /*contextual menu pointer*/
   NewAccess(CATCmdContainer, p_contextual_menu, TcCtxMenuAddin);

   /*add-in and command for Load Merge Selected Level*/
   NewAccess        (CATCmdStarter, p_command1, command1);
   SetAccessCommand (p_command1, "Cat2Load_merge_SelectedLevel");
```

```
AddAccessChild   (p_contextual_menu, p_command1);

/*add-in and command for Load Merge Target*/
NewAccess        (CATCmdStarter, p_command2, command2);
SetAccessCommand (p_command2, "Cat2Load_merge_Target");
SetAccessNext    (p_command1, p_command2);

/*add-in and command for Read Linked Document*/
NewAccess        (CATCmdStarter, p_command3, command3);
SetAccessCommand (p_command3, "Cat2Read_linked_documents");
SetAccessNext    (p_command2, p_command3);

/*separator*/
NewAccess        (CATCmdSeparator, p_separator1, separator1);
SetAccessNext    (p_command3, p_separator1);

/*add-in and command for Insert*/
NewAccess        (CATCmdStarter, p_command4, command4);
SetAccessCommand (p_command4, "Cat2Insert");
SetAccessNext    (p_separator1, p_command4);

/*add-in and command for Replace*/
NewAccess        (CATCmdStarter, p_command5, command5);
SetAccessCommand (p_command5, "Cat2Replace");
SetAccessNext    (p_command4, p_command5);

/*add-in and command for ReplaceByRevision*/
NewAccess        (CATCmdStarter, p_command6, command6);
SetAccessCommand (p_command6, "Cat2ReplaceByRevision");
SetAccessNext    (p_command5, p_command6);

/*separator*/
NewAccess        (CATCmdSeparator, p_separator2, separator2);
SetAccessNext    (p_command6, p_separator2);

/*add-in and command for Replace Load As Cgr*/
NewAccess        (CATCmdStarter, p_command7, command7);
SetAccessCommand (p_command7, "Cat2ReplaceLoadAsCgr");
SetAccessNext    (p_separator2, p_command7);

/*separator*/
NewAccess        (CATCmdSeparator, p_separator3, separator3);
SetAccessNext    (p_command7, p_separator3);

/*add-in and command for Check-in*/
NewAccess        (CATCmdStarter, p_command8, command8);
SetAccessCommand (p_command8, "Cat2Check_In_Selection");
SetAccessNext    (p_separator3, p_command8);

/*add-in and command for Check-out*/
NewAccess        (CATCmdStarter, p_command9, command9);
SetAccessCommand (p_command9, "Cat2Check_Out_Selection_Ctx");
```

```
   SetAccessNext    (p_command8, p_command9);

   /* add-in and commande for statesDetailed*/
   NewAccess        (CATCmdStarter, p_command10, command10);
   SetAccessCommand (p_command10, "Cat2DetailedStates");
   SetAccessNext    (p_command9, p_command10);

   /* add-in and command for Highlight in Teamcenter */
   NewAccess        (CATCmdStarter, p_command11, command11);
   SetAccessCommand (p_command11, "Cat2HighlightInTc");
   SetAccessNext    (p_command10, p_command11);

   /*separator*/
   NewAccess        (CATCmdSeparator, p_separator4, separator4);
   SetAccessNext    (p_command6, p_separator4);

  /* add-in and command for MyCommand_Custo */
   NewAccess        (CATCmdStarter, p_command12, command12);
   SetAccessCommand (p_command12, "MyCommand_Custo");
   SetAccessNext    (p_command11, p_command12);

   /*separator*/
   NewAccess        (CATCmdSeparator, p_separator5, separator5);
   AddAccessChild   (o_contextual_menu, p_separator5);

   /*set the contextual menu title*/
   contextual_menu_title =
CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
              "CV5CAAToolbarHeader.TcCtxMenu");
   p_contextual_menu->SetTitle(contextual_menu_title);
   SetContextualMenuTitle();

   /*set the contextual menu*/
   SetAccessNext(p_separator5, p_contextual_menu);

   return S_OK;
}

/*Set the titles to the Contextual Menu*/
HRESULT CTXMenuProductAddin::SetContextualMenuTitle()
{
   CATUnicodeString menu_title;
   CATCommandHeader *p_command = NULL;
   /*SetTitle to Load Merge Selected Level*/
   ::CATAfrGetCommandHeader("Cat2Load_merge_SelectedLevel",
p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Load_merge_SelectedLevel.S
hortHelp");
   p_command->SetTitle(menu_title);

   /*SetTitle to Load Merge Target*/
   ::CATAfrGetCommandHeader("Cat2Load_merge_Target", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Load_merge_Target.ShortHel
p");
   p_command->SetTitle(menu_title);

   /*SetTitle to Read Linked Document*/
   ::CATAfrGetCommandHeader("Cat2Read_linked_documents",
p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Read_linked_documents.Shor
tHelp");
   p_command->SetTitle(menu_title);

   /*SetTitle to Insert*/
   ::CATAfrGetCommandHeader("Cat2Insert", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Insert.ShortHelp");
   p_command->SetTitle(menu_title);

   /*SetTitle to Replace*/
   ::CATAfrGetCommandHeader("Cat2Replace", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Replace.ShortHelp");
   p_command->SetTitle(menu_title);

   /*SetTitle to ReplaceByRevision*/
   ::CATAfrGetCommandHeader("Cat2ReplaceByRevision", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2ReplaceByRevision.ShortHel
p");
   p_command->SetTitle(menu_title);

   /* SetTitle to Replace Load As CGR*/
   ::CATAfrGetCommandHeader("Cat2ReplaceLoadAsCgr", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
               "CV5CAAToolbarHeader.Cat2ReplaceLoadAsCgr.ShortHelp"
);
   p_command->SetTitle(menu_title);

   /*SetTitle to Check-in*/
   ::CATAfrGetCommandHeader("Cat2Check_In_Selection", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Check_In_Selection.ShortHe
lp");
   p_command->SetTitle(menu_title);

   /*SetTitle to Check-out*/
   ::CATAfrGetCommandHeader("Cat2Check_Out_Selection_Ctx",
p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2Check_Out_Selection_Ctx.Sh
ortHelp");
   p_command->SetTitle(menu_title);

   /*SetTitle to Detailed States*/
   ::CATAfrGetCommandHeader("Cat2DetailedStates", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
   "CV5CAAToolbarHeader.Cat2DetailedStates.ShortHelp");
   p_command->SetTitle(menu_title);

   /* SetTitle to Highlight in Teamcenter */
   ::CATAfrGetCommandHeader("Cat2HighlightInTc", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.Cat2HighlightInTc.ShortHelp");
   p_command->SetTitle(menu_title);

   /* SetTitle to MyCommand_Custo */
   ::CATAfrGetCommandHeader("MyCommand_Custo ", p_command);
   menu_title = CATMsgCatalog::BuildMessage("CV5CAAToolbarHeader",
                "CV5CAAToolbarHeader.MyCommand_Custo.ShortHelp");
   p_command->SetTitle(menu_title);

   return S_OK;

}
```

2. Define actions related to MyCommand_Custo command in the CATCommand::Activate() method.

```
...CATStatusChangeRC MyCommandCusto::Activate(CATCommand*
ip_from_client,
                                         CATNotification*
ip_evt_dat)
{
      CATApplication* p_application =
CATApplication::MainApplication();
      p_application->AddTimeOut(1, (CATCommand*)this, NULL,
                                              (void(*)())MyCommandC
usto::Execute);
      RequestDelayedDestruction();
      return(CATStatusChangeRCCompleted);
}
```

3. Define the resource file associated to the command in the CV5CAAToolbarHeader.CATNls and CV5CAAToolbarHeader.CATRsc files.

```
...
CV5CAAToolbarHeader.MyCommand_Custo.Icon.Normal              =
"I_04003_MyCommand_Custo";
```

```
...
CV5CAAToolbarHeader.MyCommand_Custo.Title     = "Cmd
MyCommand_Custo";
CV5CAAToolbarHeader.MyCommand_Custo.ShortHelp = "MyCommand_Custo";
CV5CAAToolbarHeader.MyCommand_Custo.Help      = "MyCommand_Custo";
CV5CAAToolbarHeader.MyCommand_Custo.LongHelp  = "MyCommand_Custo";
```

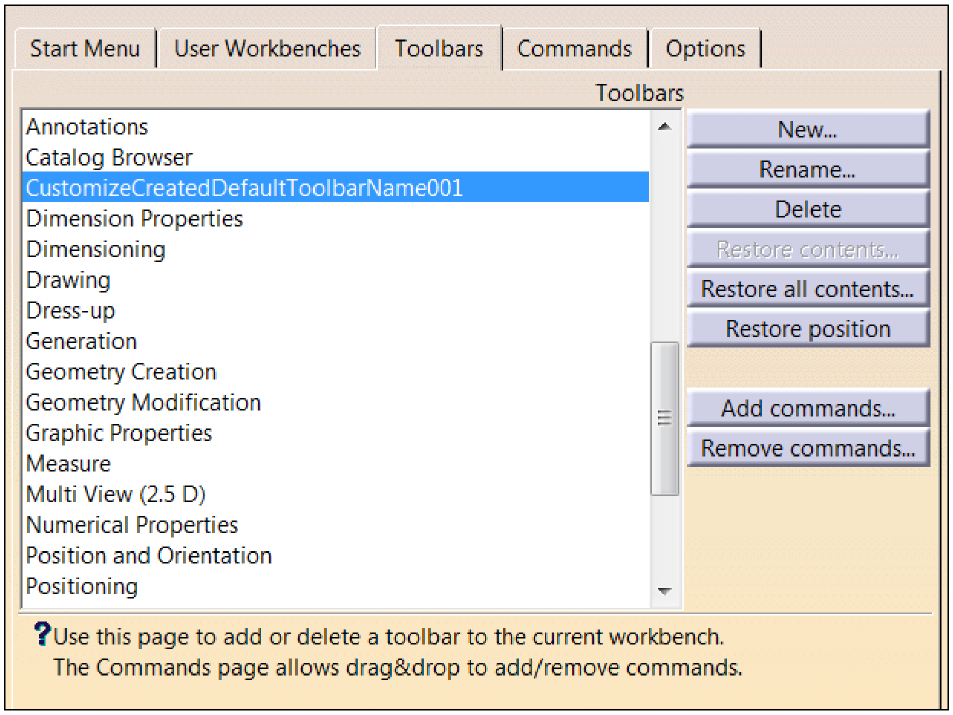
6.2.3 Create a New Toolbar

Create a Toolbar
Create a Toolbar using CAA

6.2.3.1 Create a Toolbar

Perform the following steps to manually create a toolbar.

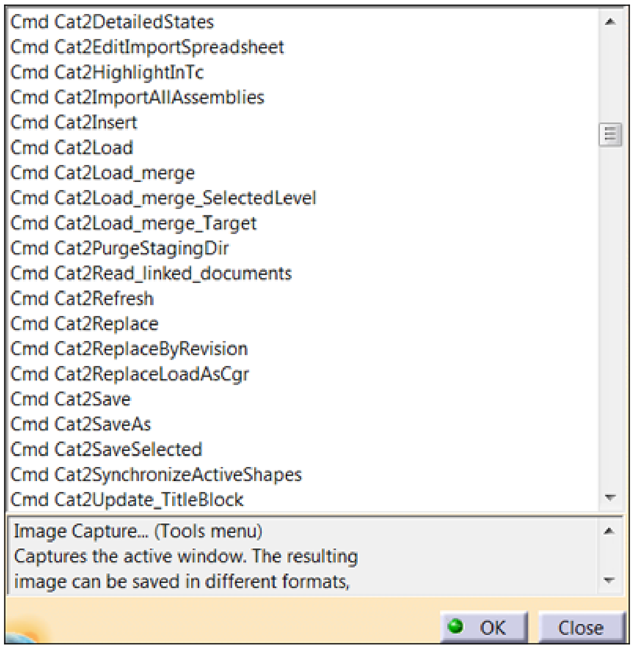
1. In CATIA, select **Tools >> Customize**, pick the **Toolbars** tab then pick the **New...** button.



Annotations
Catalog Browser
CustomizeCreatedDefaultToolbarName001
Dimension Properties
Dimensioning
Drawing
Dress-up
Generation
Geometry Creation
Geometry Modification
Graphic Properties
Measure
Multi View (2.5 D)
Numerical Properties
Position and Orientation
Positioning

?Use this page to add or delete a toolbar to the current workbench.
The Commands page allows drag&drop to add/remove commands.

2. Select the new toolbar from the list, then pick **Add Commands** to add the TcIC commands available from the list. Pick **OK** when finished.



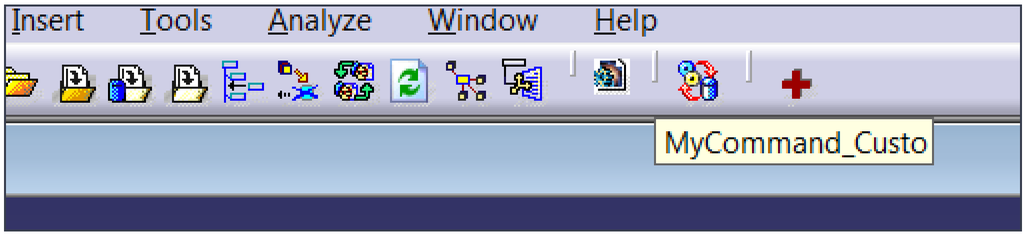
Cmd Cat2DetailedStates
Cmd Cat2EditImportSpreadsheet
Cmd Cat2HighlightInTc
Cmd Cat2ImportAllAssemblies
Cmd Cat2Insert
Cmd Cat2Load
Cmd Cat2Load_merge
Cmd Cat2Load_merge_SelectedLevel
Cmd Cat2Load_merge_Target
Cmd Cat2PurgeStagingDir
Cmd Cat2Read_linked_documents
Cmd Cat2Refresh
Cmd Cat2Replace
Cmd Cat2ReplaceByRevision
Cmd Cat2ReplaceLoadAsCgr
Cmd Cat2Save
Cmd Cat2SaveAs
Cmd Cat2SaveSelected
Cmd Cat2SynchronizeActiveShapes
Cmd Cat2Update_TitleBlock

Image Capture... (Tools menu)
Captures the active window. The resulting image can be saved in different formats,

6.2.3.2 Create a Toolbar using CAA

The toolbar can be developed in CAA when TcIC commands are only available in CAA mode.

Perform the following steps to create a new toolbar through the TIE_CATIAfrGeneralWksAddin interface implementation.



Insert an additional toolbar and commands utilizing the CreateToolbars method. In this use case, CreateToolbars method implements TLBTcUtilityAddin containing a MyCommand_Custo command.

```
 ...
/*-------------------------------------------------------------------
----------*/
#include "TIE_CATIAfrGeneralWksAddin.h"
TIE_CATIAfrGeneralWksAddin(TLBTcUtilityAddin);
...
/*-------------------------------------------------------------------
----------*/
CATCmdContainer* TLBTcUtilityAddin::CreateToolbars()
```

```
{
   /*toolbar pointer*/
   NewAccess(CATCmdContainer, p_toolbar_utility,
TLBTcUtilityAddin);
   /*add-in and command for Check-in*/
   NewAccess        (CATCmdStarter, p_command1, command1);
   SetAccessCommand (p_command1, "MyCommand_Custo");
   SetAccessChild   (p_toolbar_utility, p_command1);
   /*set the toolbar title*/
   SetAccessTitle(p_toolbar_utility, "TLBTcUtilityAddin");
   /*set (1) for a visible toolbar, (-1) for an invisible toolbar
otherwise*/
   AddToolbarView(p_toolbar_utility, 1, Top);

      return p_toolbar_utility;
}
/*------------------------------------------------------------------
----------*/
/*Set the commands to the Teamcenter_save toolbar*/
void TLBTcUtilityAddin::CreateCommands()
{
   /* Demonstrator :
    * - identifier of the command header
    * - name of the module
    * - name of the command class
    * - arguments for the command class
    */

   /*command for Check-in*/
   new CADPolToolbarHeader("MyCommand_Custo",
                           "CADPolExecution",
                           "CADMenuContextuel",
                           (void*)NULL);
}
```

3. Define the resource file associated to the command in the CADPolToolbarHeader.CATNls and CADPolToolbarHeader.CATRsc files.

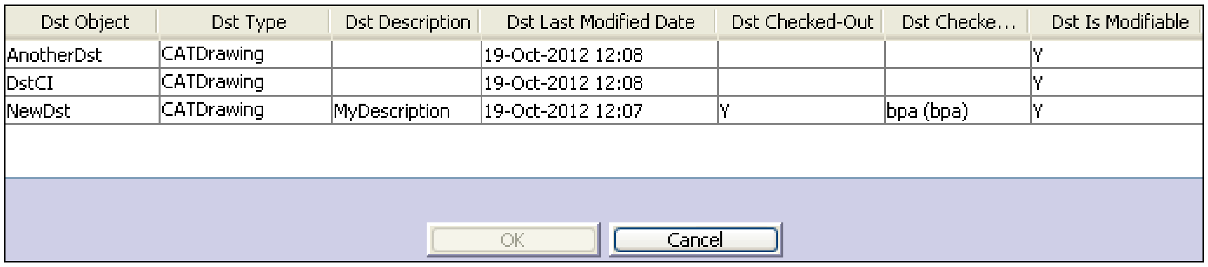
```
...
CADPolToolbarHeader.MyCommand_Custo.Icon.Normal              =
"I_04003_MyCommand_Custo";


...
CADPolToolbarHeader.MyCommand_Custo.Title     = "Cmd
MyCommand_Custo";
CADPolToolbarHeader.MyCommand_Custo.ShortHelp = "MyCommand_Custo";
CADPolToolbarHeader.MyCommand_Custo.Help      = "MyCommand_Custo";
CADPolToolbarHeader.MyCommand_Custo.LongHelp  = "MyCommand_Custo";
```

6.3 Dataset Browser

Use the **Dataset Browser** to select a dataset to use when saving a document. The Dataset Browser is available from the Save Manager and Item Revision Browser.

Modify the *$TCICV5_DIR/env/datasetbrowser.xml* file under the Save Module element to customize the Dataset Browser.

Ensure the encoding of the modified *datasetbrowser.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.



Dst Object	Dst Type	Dst Description	Dst Last Modified Date	Dst Checked-Out	Dst Checke...	Dst Is Modifiable
AnotherDst	CATDrawing		19-Oct-2012 12:08			Y
DstCI	CATDrawing		19-Oct-2012 12:08			Y
NewDst	CATDrawing	MyDescription	19-Oct-2012 12:07	Y	bpa (bpa)	Y

6.3.1 Add/Remove Columns

Add and/or remove columns within the Dataset Browser Table by modifying the **column** elements found within the `nameRef = Save Module` attribute. Each column element contains an attribute name (i.e., `<Column name="description"/>`) specifying the property to display in the column.

Remove a column from the table by removing its column element within the *datasetbrowser.xml* file. **Add** a column to the table by adding a new column element within the *datasetbrowser.xml* file.

For example, add the following lines within the *datasetbrowser.xml* file:

to add the CATIA File Name column:

```
<column name="former_file_name"/>
```

to add the Created Dataset Date column:

```
<column name="TC Dst creation_date"/>
```

6.3.2 Column Width and Titles

Adjust the width of the Database Browser Table columns by modifying the columns' **width** attribute within each column element found within the `nameRef = Save Module` attribute. The value of the column width attribute specifies the default width of the column in pixels. If no value is displayed next to the column element, the column widths automatically adjust to fit its content.

For example, to display a width of 60 pixels for the CATIA Status column, the attribute should be written as: `<Column name="TC Dst last_mod_date" width="60"/>`

6.3.3 Modify Column Titles

Modify column titles of the Dataset Browser Table by modifying the **display** attribute within each column element found within the `nameRef = Save Module` attribute.

The column titles display the value in its `display` attribute. If no value is assigned, the default localized string is used to display the title of the column depending on the `name` attribute value. The display attribute overrides the column default name, but no localization is applied to the display attribute value.

For example, to display the title, *State In CATIA,* for the catia_status column, the attribute would be written as: `<Column name="TC Dst last_mod_date" display="Last Modification Date"/>`

6.3.4 Reorder Columns

Reorder columns within the Dataset Browser by modifying the order of the **column** elements found within the `nameRef = Save Module` attribute.

The columns within the Dataset Browser Table display in the same order as displayed within the list. Rearrange the order of this list to rearrange the order of the columns within the table.

The code shown below causes the Dataset Browser to display in the following order:

1. Dst Object
2. Dst Type
3. Dst Description
4. Dst Last Modified Date
5. Dst Checked-out
6. Dst Checked-out User
7. Dst Is Modifiable

```
<!-- Columns definition -->
<Column name="TC Dst object_string"/>
<Column name="TC Dst object_type"/>
<Column name="TC Dst object_desc"/>
<Column name="TC Dst last_mod_date"/>
<Column name="TC Dst checked_out"/>
<Column name="TC Dst checked_out_user"/>
<Column name="TC Dst is_modifiable"/>
</Module>
```

6.4 Define the Environment

Customizing the CATIA_env File

The **CATIA_env** file is created during the installation of Teamcenter Integration for CATIA, and describes the environment defined during the installation (the CATIA version, installation path and environment filename).

The CATIA_env file also supports multiple CATIA environments (XML Tags <ENV>, CATIA version, installation path, environment filename and CATEnv filepath).

Environment	CATIA_env is defined within...
Windows	*%TCICV5_DIR%\env\catia_env*

UNIX	*$TCICV5_DIR/env/catia_env*

When installing Teamcenter Integration for CATIA as a **new** installation, the CATIA_env file contains only one environment as defined with the information selected during the installation.

Refer to the following information when the installation overwrites an **existing** installation the CATIA_env file.

- When the previous Integration version corresponds to a 7.0.0 (or higher) release, the CATIA_env file format is correct. A new default block (the `<ENV>..</ENV>`) contains the information selected during the InstallShield process.

  Environments similar to the default environment are not copied. For instance, if the default environment is CATIAV5_R16_CAA then all environments with the same name are not copied to the new CATIA_env file.

- When the previous Integration version does not correspond to a 7.0.0 (or higher) release, the CATIA_env file format is **not correct**. The default environment is created, and the InstallShield attempts to update this environment by detecting the CATIA_env directory and verifying the environment exists in this directory (for example CATIA_V5R17_CAA.txt). If an error occurs, the old information is not copied to the new CATIA_env file otherwise a `<ENV>..</ENV>` block is created.

6.4.1.1 Customizing the CATIA_env File

Use caution when modifying the CATIA_env file to ensure the following:

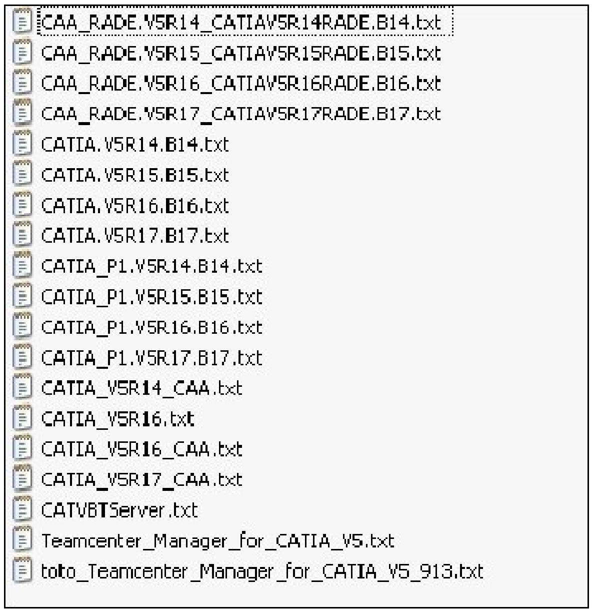
-- The first block (`<ENV>. . . .</ENV>`) is the default format.

-- Each environment must contain all four pieces of information, listed below.

The CATIA_env file must contain the following four pieces of information:

1. The CATIA V5 release (14, 15, 16, 17...).
2. The path to the CATStart.exe file: *C:\ProgramFiles\DassaultSystemes\B16\intel_a\code\bin*.
3. The name of the CATIA environment, such as, `CATIA_V5R16_CAA` which corresponds to the *CATIA_V5R16_CAA.txt* file.

4. The location of the CATIA environment file. For example: *C:\Documents and Settings\All Users\Application Data\DassaultSystemes\CATEnv*.



```
CAA_RADE.V5R14_CATIAV5R14RADE.B14.txt
CAA_RADE.V5R15_CATIAV5R15RADE.B15.txt
CAA_RADE.V5R16_CATIAV5R16RADE.B16.txt
CAA_RADE.V5R17_CATIAV5R17RADE.B17.txt
CATIA.V5R14.B14.txt
CATIA.V5R15.B15.txt
CATIA.V5R16.B16.txt
CATIA.V5R17.B17.txt
CATIA_P1.V5R14.B14.txt
CATIA_P1.V5R15.B15.txt
CATIA_P1.V5R16.B16.txt
CATIA_P1.V5R17.B17.txt
CATIA_V5R14_CAA.txt
CATIA_V5R16.txt
CATIA_V5R16_CAA.txt
CATIA_V5R17_CAA.txt
CATVBTServer.txt
Teamcenter_Manager_for_CATIA_V5.txt
toto_Teamcenter_Manager_for_CATIA_V5_913.txt
```

Each **CATIA environment** must be defined within the `<ENV></ENV>` tags. The first `<ENV>...</ENV>` block corresponds to the default environment. All entries in the CATIA_env file must be mapped to the corresponding entries defined by the CATIA_CE_EnvList preference.

For example:

```
<ENV>
16
C:\Program Files\Dassault Systemes\B16\intel_a\code\bin
CATIA_V5R16_CAA
C:\Program Files\Dassault Systemes\CATEnv
</ENV>
<ENV>
17
C:\Program Files\Dassault Systemes\B17\intel_a\code\bin
CATIA_V5R17_CAA
C:\Program Files\Dassault Systemes\CATEnv
</ENV>
<ENV>
17
C:\Program Files\Dassault Systemes\B16\intel_a\code\bin
EBS_CATIA_ENV
C:\Documents and Settings\pbu\Application Data\DassaultSystemes\CATEnv
</ENV>
```

When the **Load** command runs from Teamcenter without executing the script in CATIA, the assembly is loaded in CATIA.

When CATIA is not running, the `CNEXT` command is run with the environment defined in the CATIA_env file.

When working with **CAA** code, CATIA is run with the CAA environment (e.g. : CATIA_V5R11_CAA for CATIA V5R11) in the environment path defined during the installation of the Integration.

When working with **CATScript** code, CATIA is run with the default environment; there is no environment name and no path with which to find the environment file in the CATIA_env file.

6.5 Display Messages in the System Language

To display messages in the system language, copy the **en** directory within the messages directory and rename it using the system language name.

You must also translate the messages in both CATIA and Teamcenter directories in the appropriate language. For example:

Language	System Language Name
English	eg
French	fr
Chinese	zh

The order of messages in the files is very important. An incorrect order of messages causes them to become corrupted.

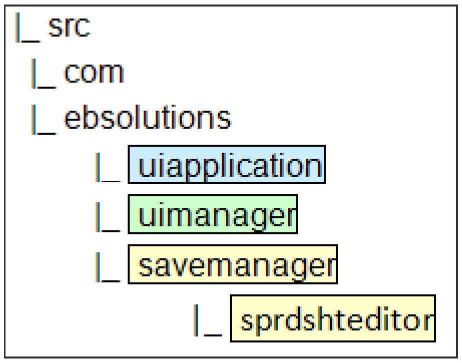
Do not use more than four consecutive '~' characters in messages because it is reserved for comments.

Do not use the '=' character in messages because it is used as separator between a message and its identifier.

6.6 Import Spreadsheet Editor

Preferences Related to the Import Spreadsheet Editor

The Import Spreadsheet Editor extends the Save Manager Application and can be customized by modifying the *$TCICV5_DIR/env/sprdshteditor.xml* file.



When the *sprdshteditor.xml* file is modified, ensure the encoding of the modified *sprdshteditor.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Refer to the Save Manager Customization section for customization information, replacing **savemanager.xml** with **sprdshteditor.xml**.

The Spreadsheet Editor panel title area can display additional information related to the current Teamcenter connection (User, Group, Role) and CATIA environment. The display format of this information is defined in the CATIA_ui_title_format preference.

When this preference is empty, information related to the current Teamcenter connection (User, Group, Role) and CATIA environment does not display.

6.6.1 Preferences Related to the Spreadsheet Editor Panel

Teamcenter must be restarted after modifying preferences.

6.6.1.1 Display Additional Information Within the Title of the Spreadsheet Editor Panel

The Spreadsheet Editor panel title area can display additional information related to the current Teamcenter connection (User, Group, Role) Current Project Name and CATIA environment. Define the title pattern displayed on the Spreadsheet Editor panel using the string below. When this preference is empty, no additional information displays.

{0} = user

{1} = group

{2} = role

{3} = name of the current CATIA environment

{4} = current project name

```
CATIA_ui_title_format = ({0}-{1}/{2} [{4}] [{3}])
```

The default value is **({0} – {1} / {2} [{4}] [{3}])** which displays as *user-group / role [current project name] [name of current CATIA environment]*.

This preference is used for the title of the Save Manager panel, Spreadsheet Editor panel and the State Details panel.

6.7 Teamcenter Integration for CATIA V5 Toolbar

Refer to **Dassault's CATIA documentation** to customize the Integration toolbar.

6.7.1 Hide All or Parts of the Toolbar

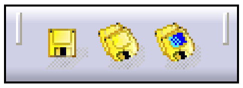
Use the Hide method available under the `CATCmdContainer` class to hide all or parts of the Integration toolbar. For example, to hide the Utility toolbar:

```
...
/*-----------------------------------------------------------------
----------*/
#include "TIE_CATIAfrGeneralWksAddin.h"
TIE_CATIAfrGeneralWksAddin(TLBTcUtilityAddin);
...
/*-----------------------------------------------------------------
----------*/
CATCmdContainer* TLBTcUtilityAddin::CreateToolbars()
{
   /*toolbar pointer*/
   NewAccess(CATCmdContainer, p_toolbar_utility,
TLBTcUtilityAddin);

   /*add-in and command for Check-in*/
   NewAccess        (CATCmdStarter, p_command1, command1);
   SetAccessCommand (p_command1, "MyCommand_Custo");
   SetAccessChild   (p_toolbar_utility, p_command1);

   /*set the toolbar title*/
   SetAccessTitle(p_toolbar_utility, "TLBTcUtilityAddin");

   /*hide the Utility Toolbar*/
   p_toolbar_utility->Hide();

 return p_toolbar_utility;
}
```

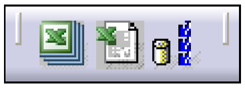
6.7.2 Icon Descriptions and File Names

Refer to the table below for the icon's description and location.

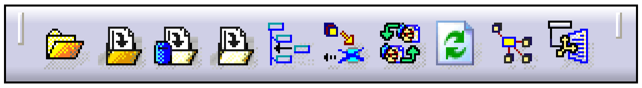
6.7.2.1 Save



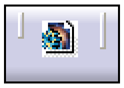
6.7.2.2 Import



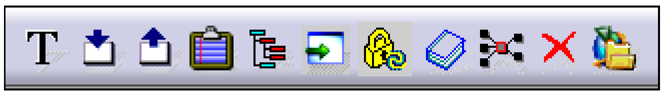
6.7.2.3 Load



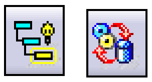
6.7.2.4 Add CATShape



6.7.2.5 Utility



6.7.2.6 Replace Load as cgr



6.7.2.7 Descriptions and File Names

Command Name	Description	Toolbar	Icon
Cat2AddCATShape (*CAA only)*	Launches the Add CATShape Shape Representations command.	Add CATShape	
Cat2CreateAllSpreadsheets or CreateAllSpreadSheets.CATScript	Creates one spreadsheet containing the data of several assemblies in order to import them.	Import	
Cat2EditImportSpreadsheet *(CAA only)*	Opens the Spreadsheet Editor in order to edit and modify the import spreadsheet	Import	
Cat2ImportAllAssemblies or ImportAllAssemblies.CATScript	Saves many assemblies originating from a supplier in Teamcenter from CATIA V5.	Import	
Cat2Load or Load.CATScript	Loads an assembly from Teamcenter to CATIA V5.	Load	
Cat2Load_merge or Load_merge.CATScript	Load Merge an assembly from Teamcenter to CATIA V5 from a selection in the PSE Portal.	Load	
Cat2Load_merge_SelectedLevel or Load_merge_SelectedLevel.CATScript	Load Merge a subassembly from Teamcenter to CATIA V5 from a selection in CATIA. Only the selected component and its children parts are exported.	Load	

Command Name	Description	Toolbar	Icon
Cat2Load_merge_Target or Load_merge_Target.CATScript	Load Merge an assembly from Teamcenter to CATIA V5 from a selection in CATIA. The selected component and all children components are exported.	Load	
Cat2Insert or Insert.CATScript	Permits elements from Teamcenter to be inserted in CATIA.	Load	
Cat2Replace or Replace.CATScript	Permits elements from Teamcenter to be replaced in CATIA.	Load	
Cat2ReplaceByRevision or Cat2ReplaceByRevision.CATScript	Replaces the element selected in CATIA by the corresponding Revision stored in Teamcenter	Load	
Cat2Refresh or Refresh.CATScript	Reloads a selected checked-in subassembly or component in CATIA.	Load	
Cat2Read_linked_documents or Read_linked_documents.CATScript	Reads the linked documents of the components from the active window in CATIA V5 from Teamcenter to CATIA V5.	Load	
Cat2SynchronizeActiveShapes *(CAA only)*	Launches the Synchronize Active Shape Representations command.	Load	

Command Name	Description	Toolbar	Icon
Cat2ActivateTerminalNodes (CAA only)	Export cache files in CATIA on an assembly loaded as reference with an assembly configuration	Replace Load as cgr	
Cat2ReplaceLoadAsCgr or Cat2ReplaceLoadAsCgr.CATScript	Replaces a CGR component with the corresponding CATPart from Teamcenter.	Replace Load as cgr	
Cat2Save *(CAA only)*	Saves an assembly in Teamcenter from CATIA V5. The Save mode is selected from a panel in the Portal.	Save	
Cat2SaveAs *(CAA only)*	Saves the selected components as new, existing or revised in Teamcenter from CATIA V5.	Save	
Cat2SaveSelected *(CAA only)*	Saves a selected component as existing or as new revision in Teamcenter from CATIA V5. For a subassembly, it saves only the first level of children.	Save	
Cat2Teamcenter or Teamcenter.CATScript	Displays the Teamcenter window.	Teamcenter Menu	
Cat2Documentation *(CAA only)*	Displays the Teamcenter Integration for CATIA V5 Documentation	Teamcenter Menu	

Command Name	Description	Toolbar	Icon
Cat2UserSettings *(CAA only)*	Connects to the RACLess Application to change Group, Role and Project of the current RACLess Application	Teamcenter Menu	👤
Cat2AboutIntegration *(CAA only)*	Displays detailed information related to the Teamcenter Integration for CATIA V5 version installed	Teamcenter Menu	---
Cat2Update_TitleBlock or Update_TitleBlock.CATScript	Updates the title block of a drawing in CATIA V5 from Teamcenter.	Utility	T
Cat2Check_In_Selection (*CAA only)*	Checks-in the selected references in Teamcenter.	Utility	📥
Cat2Check_Out_Selection (*CAA only)*	Checks-out the selected references in Teamcenter.	Utility	📤
Cat2DetailedStates (*CAA only)*	Displays status information about elements loaded into CATIA.	Utility	📋
Cat2UpdateStatus (*CAA only)*	Displays and/or updates the savable status of selected documents in the CATIA specification tree. It also displays the configurable status.	Utility	🔒

Command Name	Description	Toolbar	Icon
Cat2HighlightInTc	Highlight occurrences of documents which are selected in the CATIA specification tree, in the Teamcenter window of the Structure Manager.	Utility	
Cat2ShowInTeamcenter	Show in Teamcenter	Utility	
Cat2BrowseMML or BrowseMML.CATScript	Launches the Multi-model Links Browser.	Utility	
Cat2PurgeStagingDir or PurgeStagingDir.CATScript	Purges the staging directory.	Utility	✖

6.8 Item Revision Browser

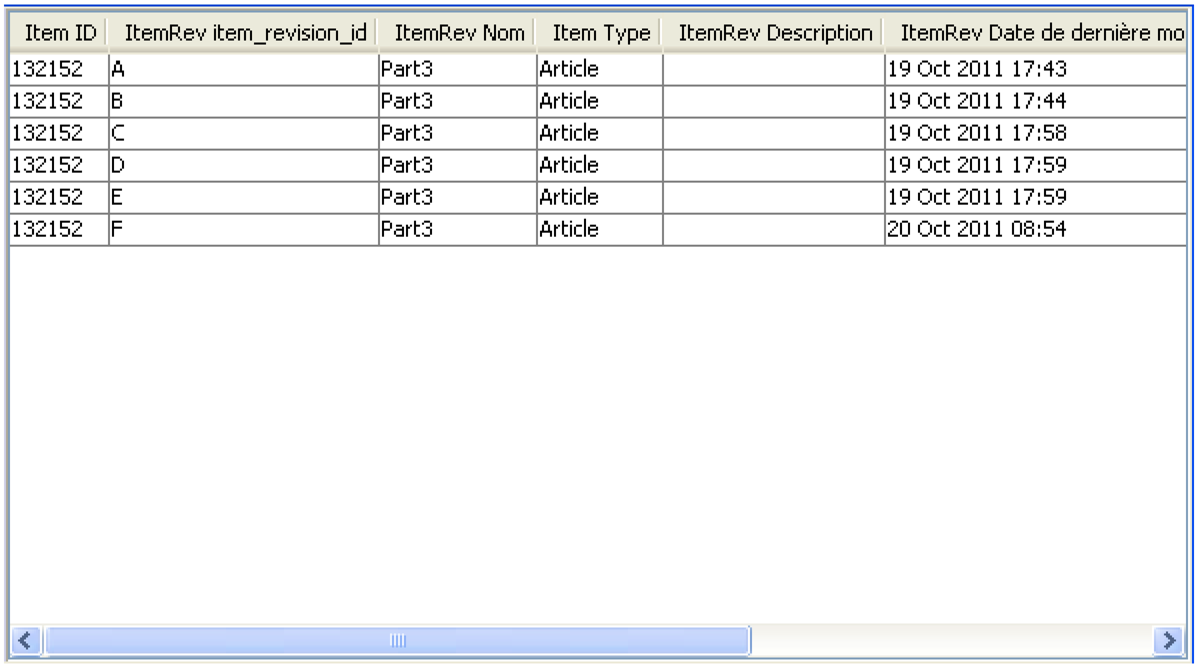
The **Item Revision Browser** displays when the **Search** Search button is selected from the Save Manager panel or when a selected element is not found in the database during automatic Save as New Revision, Use Existing Item, Use Existing Dataset operations.

Use the Item Revision Browser from the Save Manager panel to search for and select an item revision from the database to be used during the New Revision, Use Existing Item and Use Existing Dataset processes. Search results display in the query results table and the Save Manager uses the selected item.

Refer to the queries import documentation if the itemrevbrowser_query.xml has not been imported into Teamcenter before modifying the search query.

Search results display in the search query table of the Item Revision Browser. Modify the *$TCICV5_DIR/env/**itemrevbrowser.xml*** file in the Save Module/Module element to customize this table.

Queries that do not return an item or an item revision are not supported and will not display in the Item Revision Browser.



Item ID	ItemRev item_revision_id	ItemRev Nom	Item Type	ItemRev Description	ItemRev Date de dernière mo
132152	A	Part3	Article		19 Oct 2011 17:43
132152	B	Part3	Article		19 Oct 2011 17:44
132152	C	Part3	Article		19 Oct 2011 17:58
132152	D	Part3	Article		19 Oct 2011 17:59
132152	E	Part3	Article		19 Oct 2011 17:59
132152	F	Part3	Article		20 Oct 2011 08:54

6.8.1 Add/Remove Columns

Refer to the queries import documentation if the itemrevbrowser_query.xml has not been imported into Teamcenter before modifying the search query.

Add and/or remove columns within the search query table by modifying the **column** elements found within the `nameRef = Save Module` attribute. Each column element contains an attribute name (i.e., `<Column name="item_id"/>`) specifying the property to display in the column.

Remove a column from the table by removing its column element within the itemrevbrowser.xml file. **Add** a column to the table by adding a new column element within the itemrevbrowser.xml file. Refer to the following example, if necessary.

For example, add the following lines within the itemrevbrowser.xml file:

to add the CATIA File Name column:

```
<column name="former_file_name"/>
```

to add the Created Dataset Date column:

```
<column name="TC Dst Date Created"/>
```

to add the item comment of the ItemRev Master Form:

```
<column name="TC ItemRevMaster item_comment"/>
```

to add a CATIA description:

```
<column name="CATIA description"/>
```

6.8.2 Adjust Column Widths

Refer to the queries import documentation if the itemrevbrowser_query.xml has not been imported into Teamcenter before modifying the search query.

Adjust column widths of the search query table by modifying the columns' **width** attribute within each column element found within the nameRef = Save Module attribute.

The value of the column width attribute specifies the default width of the column in pixels. If no value is displayed next to the column element, the column widths automatically adjust to fit its content.

The following example indicates the CATIA Status column displays a width of 60 pixels.

```
<Column name="TC Item object_type" width="100"/>
```

6.8.3 Modify Column Titles

Refer to the queries import documentation if the itemrevbrowser_query.xml has not been imported into Teamcenter before modifying the search query.

Modify column titles of the search query table by modifying the display attribute within each column element found within the nameRef = Save Module attribute.

The column titles display the value in its display attribute. If no value is assigned, the default localized string is used to display the title of the column depending on the name attribute value. The display attribute overrides the column default name, but no localization is applied to the display attribute value.

The example below indicates the catia_status column displays State In CATIA.

```
<Column name="TC ItemRev object_name" display="Name"/>
```

The display attribute does not interact with the editor property line corresponding to this table column.

6.8.4 Reorder Columns

Refer to the queries import documentation if the itemrevbrowser_query.xml has not been imported into Teamcenter before modifying the search query.

Reorder columns within the search query table by modifying the order of the **column** elements within the *$TCICV5_DIR/env/itemrevbrowser.xml* file in the Save Module/Module element .

The columns display in the same order as displayed within the list. Rearrange the order of this list to rearrange the order of the columns within the table.

For example, columns in the Item Revision Browser table shown above are listed in the following order:

- Item ID
- Item Revision ID
- Item Revision Object Name
- Item Object type

- Item Revision Object Description
- Item Revision Last modified date

```
<!-- SAVE MODULE -->
<Module nameRef="IB Module"
…

   <!-- Columns definition -->
                     <Column name="TC Item item_id"/>
                     <Column name="TC ItemRev item_revision_id"/>
                     <Column name="TC ItemRev object_name"/>
                     <Column name="TC Item object_type"/>
                     <Column name="TC ItemRev object_desc"/>
                     <Column name="TC ItemRev last_mod_date"/>
                     </Module>
   </UIApplication>
```

6.9 JT: The catia2jt Script

There are two main steps in the JT conversion process:

1. Create the JT datasets.
2. Convert and import JT files.

The catia2jt script must be modified prior to converting CATIA V5 documents into JTs, unless the Dispatcher Services method is used.

Example

The translator batch file uses three parameters:

1. The first parameter is the converting type.
   - 0 = convert a product file
   - 1 = convert a directory of V4 files, else convert a directory of V5 files
2. The second parameter is the full path of the file name (if the first parameter = 1) or the input directory.

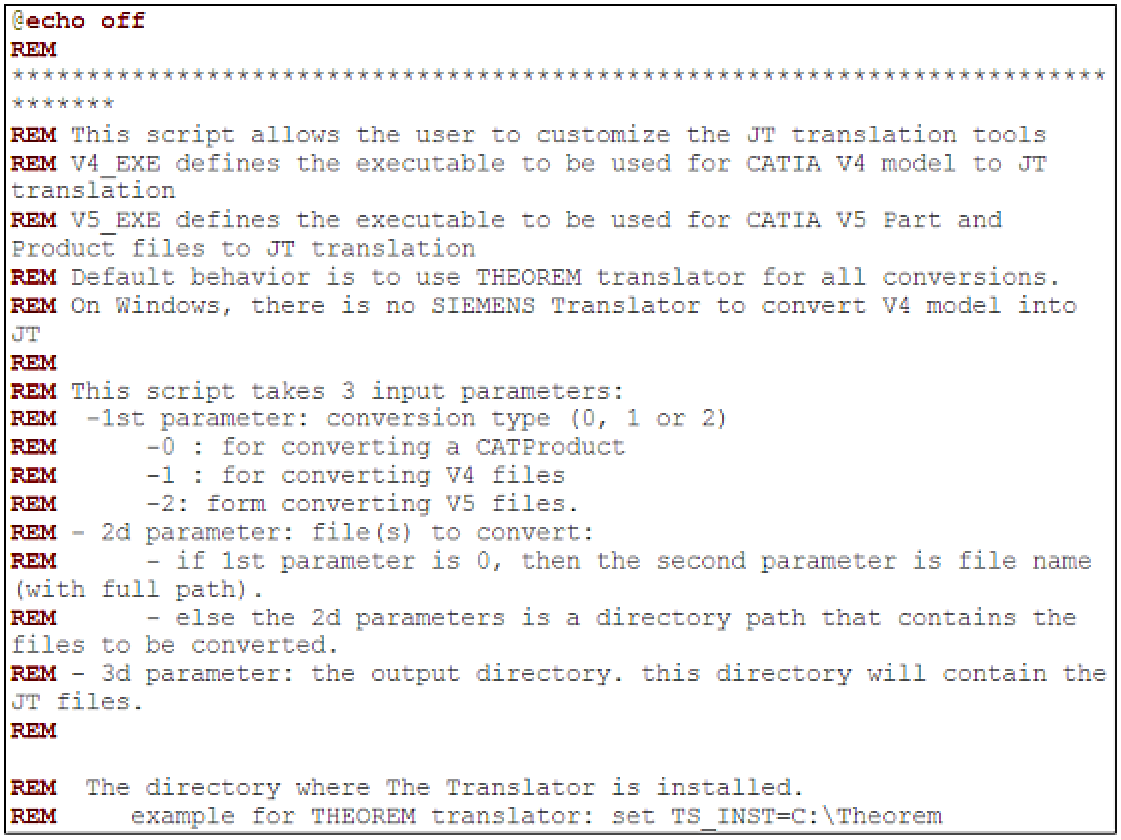
3. The third parameter is the output directory.

The illustration below was created by setting:

- **TS_INST**: The directory where the translator is installed (Theorem, by default).

- **V5_EXE**: The path to the executable file which converts V5 and CATProducts files (Theorem, by default). If the executable is not the same for V5 and CATProduct files, set the path in the command labeled "PRODUCT:" for CATProduct and "V5:" for V5 files. The configuration file is specified in the command for each type.

- **V4_EXE**: The path to the executable file which converts V4 files (Theorem, by default).



```
@echo off
REM
***************************************************************************
*******
REM This script allows the user to customize the JT translation tools
REM V4_EXE defines the executable to be used for CATIA V4 model to JT
translation
REM V5_EXE defines the executable to be used for CATIA V5 Part and
Product files to JT translation
REM Default behavior is to use THEOREM translator for all conversions.
REM On Windows, there is no SIEMENS Translator to convert V4 model into
JT
REM
REM This script takes 3 input parameters:
REM  -1st parameter: conversion type (0, 1 or 2)
REM      -0 : for converting a CATProduct
REM      -1 : for converting V4 files
REM      -2: form converting V5 files.
REM - 2d parameter: file(s) to convert:
REM      - if 1st parameter is 0, then the second parameter is file name
(with full path).
REM      - else the 2d parameters is a directory path that contains the
files to be converted.
REM - 3d parameter: the output directory. this directory will contain the
JT files.
REM

REM  The directory where The Translator is installed.
REM     example for THEOREM translator: set TS_INST=C:\Theorem
```

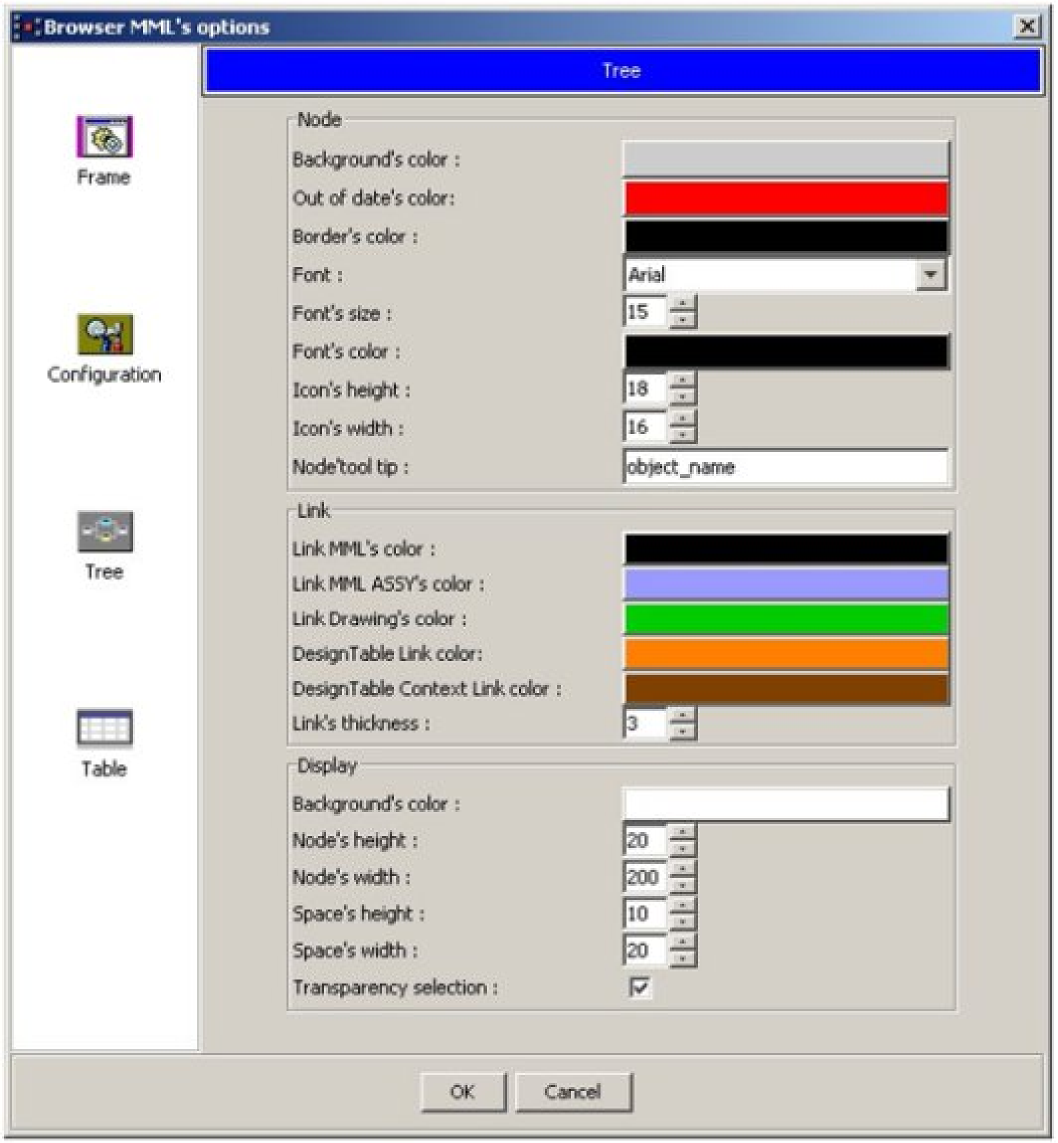
6.10 Customize the MML Browser Display

6.10.1 Options

The Multi-Model Links Browser is fully customizable. Pick **File >> Options** to customize the display by modifying the following features.

- Frame size.
- The Configuration default to specify the default display features.
- The Tree option which includes the node, links and panel colors and dimensions.

- The Table option which includes the properties displayed within the table.



Node

Background's color :	
Out of date's color:	🟥
Border's color :	⬛
Font :	Arial
Font's size :	15
Font's color :	⬛
Icon's height :	18
Icon's width :	16
Node'tool tip :	object_name

Link

Link MML's color :	⬛
Link MML ASSY's color :	🟪
Link Drawing's color :	🟩
DesignTable Link color:	🟧
DesignTable Context Link color :	🟫
Link's thickness :	3

Display

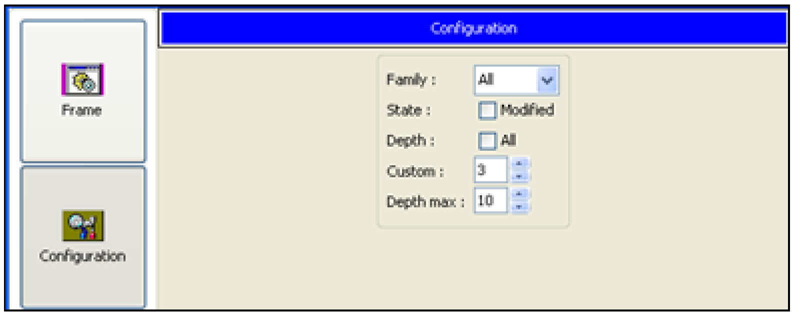
Background's color :	⬜
Node's height :	20
Node's width :	200
Space's height :	10
Space's width :	20
Transparency selection :	✓

6.10.1.1 Frame

Use the **Frame** option to specify the default height and width (in pixels) used when the application is launched, and the historic size (number of sessions to store).

6.10.1.2 Configuration

Use the **Configuration** option to specify the default setting for the following features.



Configuration

Family : All
State : ☐ Modified
Depth : ☐ All
Custom : 3
Depth max : 10

Name	Icon
Family	
State	
Depth	☐ All
Custom	3
Depth max	Maximum number of links to consider (for browsing and display)

6.10.1.3 Tree

Use the **Tree** option to specify the display of nodes, links and the layout of different components. Use the color choosers to create a default color and text fields and spinners for sizes.

Use the **Transparency selection** option to display/hide the transparent rectangle which displays as nodes are selected. It may be a useful feature, but it is resource consuming. Refer to Managing Properties section to indicate the property displayed in the nodes' tooltip.

6.10.1.4 Table

Use the **Table** option to modify, add or remove properties displayed in the table view.

6.10.2 Managing Properties

Any Teamcenter property can be used as the source of what displays in the node's tooltip or in the table columns. The property's real name should be used as the source (not the display name). Teamcenter properties can include:

- the item revision of the dataset displayed in the browser
- the dataset
- Teamcenter file related to the dataset.

For example, to display the Object Name in the table, add `object_name` in the **Identifier** and `Object Name` in the **Displayed Name** within the Table option of the Browser Customization Panel. The Browser will first search for a property named `object_name` in the dataset, and if not found, will search within the Teamcenter file.

The item revision, dataset, or file can also be displayed by adding the following prefixes to the property.

- `REV.` - Item revision properties
- `DST.` - Dataset properties
- `FILE.` - Teamcenter file properties.

6.10.3 Reset the Browser Display

Delete the file containing your local settings to cancel your customizations. Depending on your installation, this file may be located in the following directory.

Save a copy of this file in a safe place when re-installing the Integration in order to save any customizations. After installation, copy the file back to its original place.

```
TcIC_config_dir = C:\tmp
```

If **TCICV5_DIR = C:\catiman**, then the directory name is *C-_catiman* and the bmml.user.properties path is *C:\tmp\.TcIC\C-_catiman\bmml_user.properties.*

6.10.4 Customize the MML Browser Display: Common Errors

Error	Solution
UNKNOWN PROP incorrectly displays in the node's tooltip or in a table column.	Ensure the property real name is spelled correctly then verify the property exists. Consult Teamcenter documentation to locate the correct property to display.
The message *There should not be empty fields* displays on the customization dialog box after selecting **OK**.	Ensure fields are completely correctly.
The `java.lang.NumberFormatException` message displays in Teamcenter's console when attempting to launch the MML Browser from CATIA V5 or Teamcenter.	The error message may display when local settings have been changed. Reset the Browser Display then restart Teamcenter.

6.11 Modify the Look & Feel

When a pse_user.properties file is used to configure the Teamcenter Integration for CATIA menu, it is necessary to **add** the following entries to this file:

```
com.ugsolutions.iman.pse.PSEApplication.MENUBAR=com.ebsolutions.cat
iman.CatiaPSEApplicationMenuBar
catiaV5_loadAction=com.ebsolutions.catiman.actions.LoadAction
catiaV5_loadAction.NAME=Load in CATIA V5

catiaV5_calculateAssemblyAction=com.ebsolutions.catiman.actions.Cal
culateAssemblyAction
catiaV5_calculateAssemblyAction.NAME=Update assembly mass
properties

catiaV5_calculateAbsoluteAction=com.ebsolutions.catiman.actions.Cal
culateAbsoluteAction
catiaV5_calculateAbsoluteAction.NAME=Display mass properties in the
top assembly origin

catiaV5_returnToCatiaActionV5=com.ebsolutions.catiman.actions.Retur
nToCatiaAction
catiaV5_returnToCatiaActionV5.NAME=Return to CATIA V5

catiaV5_volumeSearchAction=com.ebsolutions.catiman.actions.VolumeSe
archAction
catiaV5_volumeSearchAction.NAME=Volume search

catiaV5_cat_PV_catia2jtAction=com.ebsolutions.catiman.actions.CatPV
Catia2JtAction
catiaV5_cat_PV_catia2jtAction.NAME=Convert into JT file

catiaV5_ebsAboutAction=com.ebsolutions.catiman.actions.EBSAboutActi
on
catiaV5_ebsAboutAction.NAME=About integration

catiaV5_loadConsultingAction=com.ebsolutions.catiman.actions.LoadCo
nsultingAction
catiaV5_loadConsultingAction.NAME=Load for consulting in CATIA V5

catiaV5_createCatiaPartAction=com.ebsolutions.catiman.actions.Creat
eCatiaPartAction
catiaV5_createCatiaPartAction.NAME=Create CATPart dataset for
selection
catiaV5_createCatiaPartAction.COMMAND=catiaV5_createCatiaPartComman
d
catiaV5_createCatiaPartCommand=com.ebsolutions.catiman.commands.Cre
ateDatasetCommand
```

```
catiaV5_createCatiaAssemblyAction=com.ebsolutions.catiman.actions.C
reateCatiaAssemblyAction
catiaV5_createCatiaAssemblyAction.NAME=Create CATProduct dataset
for selection
catiaV5_createCatiaAssemblyAction.COMMAND=catiaV5_createCatiaAssemb
lyCommand
catiaV5_createCatiaAssemblyCommand=com.ebsolutions.catiman.commands
.CreateDatasetCommand

catiaV5_loadSelectedLevelAction=com.ebsolutions.catiman.actions.Loa
dSelectedLevelAction
catiaV5_loadSelectedLevelAction.NAME=Load selected level in CATIA
V5

catiaV5_exportAssemblyFromPSEAction=com.ebsolutions.catiman.actions
.ExportAssemblyFromPSEAction
catiaV5_exportAssemblyFromPSEAction.NAME=Export

catiaV5_createSprdSheetAction=com.ebsolutions.catiman.actions.Creat
eSprdSheetAction
catiaV5_createSprdSheetAction.NAME=Create export spreadsheet

catiaV5_updateOEMFormsAction=com.ebsolutions.catiman.actions.Update
OEMFormsAction
catiaV5_updateOEMFormsAction.NAME=Update OEM import forms

catiaV5_browseMMLAction=com.ebsolutions.catiman.actions.BrowseMMLAc
tion
catiaV5_browseMMLAction.NAME=Browse multi-model links

catiaV5_loadForCompare=com.ebsolutions.catiman.actions.LoadForCompa
reAction
catiaV5_loadForCompare.NAME=Load for compare in CATIA V5

catiaV5_activate_latestActiveShapes.NAME=Activate latest active
shapes
catiaV5_activate_latestActiveShapes=com.ebsolutions.catiman.actions
.ActivateLatestActiveShapesAction

catiaV5_backgroundRoundRobin=com.ebsolutions.catiman.actions.Backgr
oundRoundRobinFromPSEAction
catiaV5_backgroundRoundRobin.NAME=Load, update, save
```

6.12 Save Manager

The Save Manager displays when a CATIA user saves an assembly in Teamcenter using the Integration macros. The Save Manager collects information required during the Save process and can be customized.

6.12.1 The Save Manager Editor Panel

The Editor Panel cannot customized by the .xml file. Properties in this panel are managed by the core of the interface. However, Item ID, Revision and other similar properties can be customized.

Perform the following steps to customize specific properties within the Editor Panel.

1. Use the `jar xvf Cat2uimanager.jar` command to unjar the Cat2uimanager.jar file. Note a JDK toolkit is required to access the jar command.

2. Locate the **modules_locale.properties** and **modules_locale_fr.properties** files within the **Cat2uimanager\com\ebsolutions\uimanager\modules** package. These files contain localization for every Editor property.

3. Modify lines within the file to change names of the properties. For example,

    a. Locate the **modules_locale_<yourlocale>.properties** file.

    b. Replace the `save_module.item_id.NAME=`**`Item Id`** line with `save_module.item_id.NAME=`**`Part Number`**.

4. Re-jar the Cat2uimanager.jar file.

5. Place the shell in the root directory of the original file location.

6. Enter `jar cvf Cat2uimanager.jar com WaitForSemaph.class`.

7. Replace the old Cat2uimanager.jar file by the one just created. The Save Manager Editor displays the new values set in the properties file.

6.12.2 The Save Manager Table

Add/Remove/Hide Columns

Change Table Icons

Column Widths and Titles

Reorder Columns

Set the Table/Editor Divider Position

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.12.2.1 Add/Remove/Hide Columns

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Add/Remove columns within the Save Manager Table by modifying the **column** elements found within the `nameRef = Save Module` attribute. Each column element contains an attribute name (i.e., <Column name="item_id"/>) specifying the property to display in the column.

Remove a column from the table by removing its column element within the savemanager.xml file. **Add** a column to the table by adding a new column element within the savemanager.xml file. Refer to the following example, if necessary.

- Add Teamcenter properties

- Add CATIA properties (Drawing parameters).

For example, add the following lines within the savemanager.xml file:

to add the CATIA File Name column:

```
<column name="former_file_name"/>
```

to add the Created Dataset Date column:

```
<column name="TC Dst creation_date"/>
```

to add the item comment of the ItemRev Master Form:

```
<column name="TC ItemRevMaster item_comment"/>
```

to add a CATIA description:

```
<column name="CATIA description"/>
```

6.12.2.1.1 Add a New Column for Multiple CATIA Environments

Add new column in the Save Manager to display the environment read in the form, if one exists, by adding `Column name = env` within the savemanager.xml file (in catiman/env).

6.12.2.1.2 Add CATIA Properties (Drawing Parameters) as Columns

Refer to the following table to determine available properties and attributes. Modify the xml file as previously indicated.

CATIA Display Name	CATIA Real Name	Attribute
Definition	definition	CATIA definition
Description	description	CATIA description

CATIA Display Name	CATIA Real Name	Attribute
MyProperty (custom property)	MyProperty	CATIA MyProperty
Nomenclature	nomenclature	CATIA nomenclature
Part Number	partNumber	CATIA partNumber
Revision	revision	CATIA revision
Source	Source	CATIA Source

6.12.2.1.3 Add Teamcenter Properties as Columns

Modify the .xml file as previously indicated using Teamcenter Property Real Names, for example, TC Item, item_id, TCItemRev item_revision_id, TC ItemMaster object_name, etc. The format is **TC xxx yyy**.

- *xxx* is Dst, Bvr, Item, ItemRev, ItemMaster or ItemRevMaster.
- *yyy* is any attribute/property of the xxx parameter using the real name of the element.

For example, in an English environment:
TC Item item_id
TCItemRev item_revision_id
TC ItemMaster object_name

6.12.2.1.4 Column Definitions

Each `Column` element uses an attribute name to specify the property to display in the column. Available attributes are listed below.

Property name	Description
`catia_status`	Status of the component in CATIA
`co_status`	Indicates Check-out status of the current user
`document_type`	Document type
`env`	CATIA Environment used during the Save process
`file_name`	File name
`former_file_name`	Original CATIA file name
`link_type`	Link type
`save_mode`	Save mode
`status`	Component status

6.12.2.1.5 *Hide Columns*

Use the `visible` attribute to hide columns in the Save Manager. For example, `<Column name="catia_status"` **`visible="false"`**`/>` hides the Status column.

6.12.2.2 Change Table Icons

Perform the following steps to change the Save Manager Table icons.

1. Uncompress the Cat2tciccommon.jar file by using the `jar xvf Cat2tciccommon.jar` command. Note a JDK toolkit is necessary to access the jar command.
2. Locate the default Save Manager Table icons within the **com.ebsolutions.savemanager.modules.images** package. The icon files are in .png .gif and .jpg formats.
3. Replace the existing icons as needed. For example, replace the product.png file with a new icon file to replace the current default CATProduct icon.
4. When finished, re-jar the Cat2tciccommon.jar file.
5. Place the shell into the root directory of the original file.
6. Enter the `jar cvf Cat2tciccommon.jar com WaitForSemaph.class` command to generate a new jar file containing new icons.
7. Replace the old Cat2tciccommon.jar file with the new one just created.
8. The Save Manager Table displays new icons.

The icons contained in the package are also used for other Save Manager components such as action buttons. Ensure the correct icon files are modified. By default, the icon size should be 25x25 pixels to fit the Save Manager display.

6.12.2.3 Column Width and Titles

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.12.2.3.1 Adjust Column Width

Adjust the width of the Save Manager Table columns by modifying the columns' **width** attribute within each column element found within the `nameRef = Save Module` attribute. The value of the column **width** attribute specifies the default width of the column in pixels. If no value is displayed next to the column element, the column widths automatically adjust to fit its content.

For example, to display a width of 60 pixels for the CATIA Status column, the attribute should be written as: `<Column name="catia_status" width="60"/>`

6.12.2.3.2 Change Column Titles

Modify column titles of the Save Manager Table by modifying the **display** attribute within each column element found within the `nameRef = Save Module` attribute.

The column titles display the value in its `display` attribute. If no value is assigned, the default localized string is used to display the title of the column depending on the `name` attribute value. The display attribute overrides the column default name, but no localization is applied to the display attribute value.

For example, to display the title,*State In CATIA,* for the catia_status column, the attribute would be written as: `<Column name="catia_status" display="State In CATIA"/>`

The `display` attribute does not interact with the editor property line corresponding to this table column. Refer to the Editor Customization section for further details on customizing the Editor.

6.12.2.4 Reorder Columns

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Reorder columns within the Save Manager Table by modifying the order of the **column** elements found within the `nameRef = Save Module` attribute.

The columns within the Save Manager Table display in the same order as displayed within the list. Rearrange the order of this list to rearrange the order of the columns within the table.

The code shown below causes the Save Manager Table to display in the following order:

1. Item ID
2. Item Revision ID
3. Type
4. Status
5. CATIA Status
6. CO Status
7. Save Mode

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 ...
 <UIApplication class="com.ebsolutions.savemanager.Application"
              ...>
...
   <!-- SAVE MODULE -->
   <Module nameRef="Save Module"
       name="save_module"
       ...>
     <!-- Columns definition -->
     <Column name="status"/>
     <Column name="catia_status" width="60"/>
     <Column name="save_mode"/>
     <Column name="TC item item_id" width="225"/>
```

```
      <Column name="TC itemRev item_revision_id"/>
      <Column name="document_type"/>
      <Column name="TC Item object_type"/>
      <Column name="co_status" width="29"/>

      ...

    </Module>

...

</UIApplication>
```

6.12.2.5 Set the Table/Editor Divider Position

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Determine the position of the table/editor divider in the Save Manager by modifying the **Divider** element within the within the `nameRef = Save Module` attribute. The **Divider** element is at the same level as the **Column** attribute, contains the **location** attribute. The value of the **location** attribute corresponds to a percentage between the left and right side of the Save Manager.

For example, set the Divider location attribute to .02 so that the default location of the table/editor is 20% from the left side of the Save Manager; `<Divider location="0.2"/>`

6.12.3 The Save Manager Toolbar

Customize the Right Vertical Toolbar
Customize the Top Horizontal Toolbar

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.12.3.1 Customize the Right Vertical Toolbar

Change the Toolbar Order

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.12.3.1.1 Change the Toolbar Order

Ensure to Copy/Paste the entire block of the toolbar definition area within the .xml file, including the DisplayElements of each toolbar containing the button definition. Failure to do so may corrupt the .xml file and will prevent the Save Manager from being displayed. While changing the order of the toolbars, be careful to copy/paste the whole block of the toolbar definition in the xml file.

Change the order of the Save Manager right vertical toolbar by modifying the **Toolbar** elements found within the `nameRef = Save Module` attribute. Each **Toolbar** element defines the properties of each toolbar button within the right side of the Save Manager. Rearrange the order of this list to rearrange the order of the toolbar elements.

6.12.3.1.2 Customize Toolbar Buttons

Customize toolbar buttons within the Save Manager Table by modifying the **DisplayElement** elements within the **Toolbar** element of the `nameRef = Save Module` attribute. Each of the **DisplayElement** elements define a graphical button in the toolbar. Elements are listed in the table below.

Attribute name	Possible values	Description
`showTitle`	`true/false`	Display or hide the button title. By default, the title is not displayed but it is available as a tooltip.
`nameRef`	`any String`	This attribute overrides the default button title. The value of this attribute displays within the button's tooltip if the showTitle attribute value is False. The value of this attribute displays under the icon button if the showTitle attribute value is True.
`identifier`	`any String`	This attribute identifies the localization of the button. If the nameRef attribute is declared, this identifier is not used because nameRef will be used "as it is " and no localization proceeds.
`image`	`any String`	This attribute contains the pathname of the button's icon. This path can be a package relative path, as by default in the .xml file, or an absolute local path. Compatible icons formats are .gif, .jpg and .png.
`class`	`any String`	This attribute is designed for a developer customization, and defines the class used to render the button.

`action`	`any String`	This attribute is designed for a developer customization, and defines the action class used when the button is selected.

Every button position can be changed in every toolbar by changing the order of the **DisplayElement** elements in the corresponding toolbar definition.

Within the Integration, another button style is provided to illustrate extending the Save Manager components. For example, within the **DisplayElement** element of the right toolbar definition, replace the

`<class="com.ebsolutions.savemanager.modules.SaveModuleToolBarButton"/>`
definition with

`<class="com.ebsolutions.savemanager.modules.SaveModuleSwingToolBarButton"/>`
to create a new button style. The button now "swings" each time the mouse hovers it.

6.12.3.2 Customize the Top Horizontal Toolbar

Modify the *$TCICV5_DIR/env/savemanager.xml* file under the *SaveManager/Module* element to customize this area of the Save Manager.

When the *savemanager.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.12.3.2.1 Modify Button Icons

Supported icon formats are .png, .gif and .jpg. The .bmp format is not supported. Ensure the icons are a 25x25 pixel size to fit into the default display.

Change the icons within the Save Manager toolbar by modifying the **DisplayElement** found within the `nameRef = STATES` attribute. The Image attribute within the **DisplayElement** element defines the location of the button icons.

The value of the image attribute can be relative to a package in the classpath or can be an absolute path on the local drive. By default, icons provided with the interface are located in the .jar files, but the image attribute value can be modified to a local path.

```
<DisplayElement showTitle="false" image="c:\tmp\jt.jpg"
identifier="create_jt_flag"
class="com.ebsolutions.savemanager.modules.SaveModuleToolBarCheckBo
x" action="com.ebsolutions.savemanager.actions.CreateJTAction"/>
```

6.12.3.2.2 Modify Toolbar Order

Use caution when reordering the toolbar elements. Ensure to Copy/Paste the entire toolbar definition block, including the `DisplayElement` elements of each toolbar containing the toolbar button definition. Failure to Copy/Paste the entire information may corrupt the .xml file and prevent the Status Window from being displayed.

Modify the order of the Save Manager Table toolbar by reordering the **DisplayElement** found within `nameRef = STATES` attribute. A list of **DisplayElements** defines the Save Manager top horizontal toolbar. A graphical separator is automatically added between each toolbar element. Reorder the list of toolbar elements within **DisplayElement**, as needed, to reorder the toolbar display.

The example below indicates the Save Manager top horizontal toolbar buttons display in the following order: Compute geometric values button, Create JT button, etc.

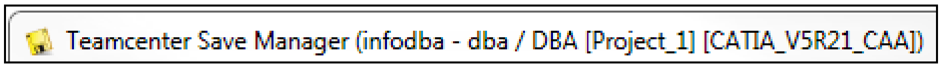
```
<?xml version="1.0" encoding="ISO-8859-1"?>
...
<UIApplication class="com.ebsolutions.savemanager.Application"
             ...>
...
 <!-- Save Manager Toolbar -->
 <SaveManagerToolBar nameRef="STATES" identifier="states"
   ...>
   <DisplayElement showTitle="false"
   image="/com/ebsolutions/uimanager/modules/images/cptgeo.gif"

   identifier="compute_geo_flag"
   class="com.ebsolutions.uimanager.modules.SaveModuleToolBarCheckB
ox"
   action="com.ebsolutions.uimanager.actions.ComputeMassPropertiesA
```

```
ction"/>

   <DisplayElement showTitle="false"
   image="/com/ebsolutions/uimanager/modules/images/createjt.png"

   identifier="create_jt_flag"
   class="com.ebsolutions.uimanager.modules.SaveModuleToolBarCheckB
ox"
   action="com.ebsolutions.uimanager.actions.CreateJTAction"/>


    ...

 </SaveManagerToolBar>

...

</UIApplication>
```

6.12.4 Save Manager Title Area

The Save Manager title area can display additional information related to the current Teamcenter connection (User, Group, Role) and CATIA Environment. The display format of this information is defined in the CATIA_ui_title_format preference.

When this preference is empty, information related to the current Teamcenter connection (User, Group, Role) and CATIA Environment does not display.


Teamcenter Save Manager (infodba - dba / DBA [Project_1] [CATIA_V5R21_CAA])

This preference affects the title of the Save Manager panel, Spreadsheet Editor panel and the State Details panel.

6.12.5 Preferences Related to the Save Manager

[Always Display the Save Manager During Save](#)

[Save a Drawing, Process or Analysis File](#)

[Use Existing Item Revision](#)

[Display Additional Information within the Title of the Save Manager Panel](#)

[How do I modify preferences and behaviors?](#)

Teamcenter must be restarted after modifying preferences.

6.12.5.1 Always Display the Save Manager During Save

Use **SAVEMANAGER_always_displayed_on_save** to display or hide the Save Manager panel during the Save Existing process on modified files, when no user interaction is required.

Set it to **true** (default) to always display the Save Manager panel containing all modified files and their parents.

Set it to **false** to hide the Save Manager panel (only if warnings or errors do not occur) during the Save Existing process. In this case, the Save Existing process proceeds without user interaction.

The Save Manager always displays if there are errors, warnings or a CATIA status set to New.

```
SAVEMANAGER_always_displayed_on_save = true
```

The default value is **true**.

6.12.5.2 Save a Drawing, Process or Analysis File

Permit the drawing, process and/or analysis file to be saved under the same item revision of the first linked document and indicate Use Existing Item as the Save Mode in the Save Manager by setting **CATIA_auto_attach_to_3d_item** to **true** (default) within Teamcenter Preferences.

Set it to **false** to permit the drawing, process and/or analysis file to be saved under a new item revision and indicate New as the Save Mode in the Save Manager.

When the drawing, process or analysis file is selected with its linked document and:

- The value of this preference is **true** then the item ID and item revision ID of the linked document is applied to the drawing, process or analysis file and the Save Mode is Existing Item.
- The value of this preference is **false** then the item ID and item revision ID are automatically assigned and the save mode is manually selected by the user.

```
CATIA_auto_attach_to_3d_item = true
```

The default value is **true**.

6.12.5.3 Use Existing Item Revision

When using the **Use Existing Item** or **Use Existing Dataset** options, automatically populate the Revision field with the latest revision available by setting the **SAVEMANAGER_use_exising_item_revision** preference to **latest**. To leave the Revision field empty (default), set this preference to **none**.

This setting must be used in combination with the SAVEMANAGER_map_item_id map.

The **Assign Revision** button can not be used to find the last revision of a specific item.

```
SAVEMANAGER_use_existing_item_revision = latest
```

The default value is **NONE**.

6.12.5.4 Display Additional Information within the title of the Save Manager Panel

The Save Manager title area can display additional information related to the current Teamcenter connection (User, Group, Role) Current Project Name and CATIA environment. Define the title pattern displayed on the Save Manager panel using the string below. When this preference is empty, no additional information displays.

{0} = user

{1} = group

{2} = role

{3} = name of the current CATIA environment

{4} = current project name

```
CATIA_ui_title_format = ({0}-{1}/{2} [{4}] [{3}])
```

The default value is **({0} – {1} / {2} [{4}] [{3}])** which displays as *user-group / role [current project name] [name of current CATIA environment]*.

This preference is used for the title of the Save Manager panel, Spreadsheet Editor panel and the State Details panel.

6.13 Customize the Seed Panel

Modify the *$TCICV5_DIR/env/seedpanel.xml* file to customize the Seed panel.

Ensure the encoding of the modified seedpanel.xml file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

By default, the *seedpanel.xml* file is configured as defined:

- TC Dst object_string
- TC Dst object_desc
- TC Dst last_mod_date

Pre-actions and post-actions are available for the **OK** and **Cancel** buttons to introduce customizations.

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
<UIApplication class="com.ebsolutions.seedpanel.SeedApplication"
       nameRef="Seed Panel"
       image="/com/ebsolutions/uimanager/modules/images/tcdesktop_1
6.png"
       sizex="" sizey="" lang="">
   <!-- MODULE -->
   <Module nameRef="Seed Module"
           name="seed_module"
           moduleHandler="com.ebsolutions.seedpanel.SeedModuleHandl
er"
           image="/com/ebsolutions/uimanager/modules/images/tcdeskt
op_16.png"
           class="com.ebsolutions.seedpanel.SeedModule">
       <!-- Columns definition -->
       <Column name="TC Dst object_string"/>
       <Column name="TC Dst object_desc"/>
       <Column name="TC Dst last_mod_date"/>
   </Module>
</UIApplication>
```

6.14 Special Characters

Preferences Related to Special Characters

Customize special characters replaced in CATIA file names by modifying the **characters_list** file located in the *$TCICV5_DIR/env* directory.

The characters already defined are characters forbidden by Windows and UNIX operating systems. It is not recommended to delete any of them. They are:

()[]<>:|?/\"*&

The list must be defined on the first line only. The characters must be written without spaces or other separators because all characters defined on the first line will be considered as special characters to be replaced.

During Import:

The Import process is stopped and an error message displays when forbidden characters, as defined within the characters_list file, are present in the Import spreadsheet

6.14.1 Preferences Related to Special Characters

Teamcenter must be restarted after modifying preferences.

Set **CATIA_replace_special_character** to change the character used to replace the special characters displayed the file name.

```
CATIA_replace_special_character = -
```

The default value is _

6.15 Status Window Customization

Pick the **DetailStates** button to display details (information from Teamcenter and CATIA) of elements loaded in CATIA from Teamcenter in the Status Window. This window can be customized.

6.15.1 Status Window Table Customization

Modify the *$TCICV5_DIR/env/statedetails.xml* file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.15.1.1 Status Window Table Customization: Add/Remove Columns

Modify the *$TCICV5_DIR/env/statedetails.xml* file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Add/Remove columns within the Status Window Table by modifying the column elements found within `nameRef = State Details Module` attribute.

Each **column element** contains an attribute name (i.e., `<Column name="item_id"/>`) specifying the property to display in the column.

Remove a column from the table by removing its column element within the statedetails.xml file. **Add** a column to the table by adding a new column element. Note that the columns display within the Status Window Table in the order listed within the .xml file. Ensure the new column is added in the correct location.

- Add Teamcenter properties
- Add CATIA properties (Drawing parameters).

For example, add the following lines within the statedetails.xml file:

to add the CATIA File Name column:

```
<Column name="former_file_name"/>
```

to add the Created Dataset Date column:

```
<Column name="TC Dst Date Created"/>
```

to add the item comment of the ItemRev Master Form:

```
<Column name="TC ItemRevMaster item_comment"/>
```

to add a CATIA description:

```
<Column name="CATIA description"/>
```

6.15.1.1.1 Column Definitions

Refer to the table below for columns available to add to the Status Window Table.

Columns can also be hidden.

Property Name	Displayed Name (EN)	Description
CATIA partNumber	CATIA partNumber	CATIA part number
catia_status	CATIA status	Status of component in CATIA: opened, modified, new, modified after synchronization
document_type	Type	CATIA Document type
env	CATIA Environment	The CATIA environment used by TcIC
former_file_name	File Name	CATIA file name
is_env_compatible	Compatible CATIA Environment	Indicates if the CATIA Environment used by TcIC is compatible with the current CATIA Environment .
is_latest_rev	Latest Revision	Indicates if the item revision of the current document is the latest item revision
modifiable_bvr	Modifiable bvr	Indicates if the bvr is modified by the current user (RW + CO)

Property Name	Displayed Name (EN)	Description
modifiable_dataset	Modifiable dataset	Indicates if the dataset is modified by the current user (RW + CO)
TC Bvr checked_out_user	Bvr Checked-Out By	Name of the user that has checked-out the bvr
TC Dst checked_out	Bvr Checked-Out	Indicates if the bvr is checked-out by the current user
TC Dst checked_out_user	Dst Checked-Out By	Name of the user that has checked-out the dataset
TC Item item_id	Item ID	Component item ID
TC ItemRev item_revision_id	ItemRev Revision	Component item revision ID
TC ItemRev release_status_list	ItemRev Release Status	Lifecycle (release) status
tc_status	Modified in Tc	true : when the current item does not exist in Teamcenter or has been modified in Teamcenter after being loaded in CATIA false: when the current item exists in Teamcenter or has not been modified in Teamcenter after being loaded in CATIA

6.15.1.1.2 Add CATIA Properties (Drawing Parameters) as Columns

Refer to the following table to determine available properties and attributes. Modify the xml file as previously indicated.

CATIA Display Name	CATIA Real Name	Attribute
Definition	definition	CATIA definition
Description	description	CATIA description
MyProperty (custom property)	MyProperty	CATIA MyProperty
Nomenclature	nomenclature	CATIA nomenclature
Part Number	partNumber	CATIA partNumber
Revision	revision	CATIA revision
Source	Source	CATIA Source

6.15.1.1.3 Add Teamcenter Properties as Columns

Modify the xml file as previously indicated using Teamcenter Property Real Names, for example, TC Item, item_id, TCItemRev item_revision_id, TC ItemMaster object_name, etc. The format is **TC xxx yyy**.

- **xxx** is Dst, Bvr, Item, ItemRev, ItemMaster or ItemRevMaster.
- **yyy** is any attribute/property of the xxx parameter using the real name of the element.

For example, in an English environment:
TC Item item_id
TCItemRev item_revision_id
TC ItemMaster object_name

Teamcenter Property	Available Attributes
BOM View Revision	TC Bvr ID ; TC Bvr Revision ; TC Bvr Name ; TC Bvr Description ; TC Bvr Type ; TC Bvr Tool Last Used ; TC Bvr Version Number ; TC Bvr Version Limit ; TC Bvr Owner ID ; TC Bvr Group ID ; TC Bvr Date Created ; TC Bvr Date Modified ; TC Bvr Last Modifying User ; TC Bvr Date Archived ; TC Bvr Date Last Backup ; TC Bvr Date Released ; TC Bvr Release Status
Dataset	TC Dst ID ; TC Dst Rev ; TC Dst Name ; TC Dst Description ; TC Dst Type ; TC Dst Tool Last Used ; TC Dst Version Number ; TC Dst Version Limit ; TC Dst Owner ID ; TC Dst Group ID ; TC Dst Date Created ; TC Dst Date Modified ; TC Dst Last Modifying User ; TC Dst Date Archived ; TC Dst Date Last Backup ; TC Dst Date Released ; TC Dst Release Status ; TC Dst k_toolLabel ; TC Dst k_typeLabel ; TC Dst k_formatLabel ; TC Dst k_siteClassLabel
Item	TC Item ID ; TC Item Revision ; TC Item Name ; TC Item Description ; TC Item Type ; TC Item Tool Last Used ; TC Item Version Number ; TC Item Version Limit ; TC Item Owner ID ; TC Item Group ID ; TC Item Date Created ; TC Item Date Modified ; TC Item Last Modifying User ; TC Item Date Archived ; TC Item Date Last Backup ; TC Item Date Released ; TC Item Release Status ; TC Item Unit of Measure
Item Master	TC ItemMaster ID ; TC ItemMaster Revision ; TC ItemMaster Name ; TC ItemMaster Description ; TC ItemMaster Type ; TC ItemMaster Tool Last Used ; TC ItemMaster Version Number ; TC ItemMaster Version Limit ; TC ItemMaster Owner ID ; TC ItemMaster Group ID ; TC ItemMaster Date Created ; TC ItemMaster Date Modified ; TC ItemMaster Last Modifying User ; TC ItemMaster Date Archived ; TC ItemMaster Date Last Backup ; TC ItemMaster Date Released ; TC ItemMaster Release Status ; TC ItemMaster Form Definition File Name

Teamcenter Property	Available Attributes
Item Revision	TC ItemRev ID ; TC ItemRev Revision ; TC ItemRev Name ; TC ItemRev Description ; TC ItemRev Type ; TC ItemRev Tool Last Used ; TC ItemRev Version Number ; TC ItemRev Version Limit ; TC ItemRev Owner ID ; TC ItemRev Group ID ; TC ItemRev Date Created ; TC ItemRev Date Modified ; TC ItemRev Last Modifying User ; TC ItemRev Date Archived ; TC ItemRev Date Last Backup ; TC ItemRev Date Released ; TC ItemRev Release Status
Item Revision Master	TC ItemRevMaster ID ; TC ItemRevMaster Revision ; TC ItemRevMaster Name ; TC ItemRevMaster Description ; TC ItemRevMaster Type ; TC ItemRevMaster Tool Last Used ; TC ItemRevMaster Version Number ; TC ItemRevMaster Version Limit ; TC ItemRevMaster Owner ID ; TC ItemRevMaster Group ID ; TC ItemRevMaster Date Created ; TC ItemRevMaster Date Modified ; TC ItemRevMaster Last Modifying User ; TC ItemRevMaster Date Archived ; TC ItemRevMaster Date Last Backup ; TC ItemRevMaster Date Released ; TC ItemRevMaster Release Status ; TC ItemRevMaster Form Definition File Name ; TC ItemRevMaster item_comment

6.15.1.1.4 Hide Columns

Use the `visible` attribute to hide columns. For example, `<Column name="catia_status" visible="false "/>` hides the CATIA Status column.

6.15.1.2 Status Window Table Customization: Change Table Icons

Modify the *$TCICV5_DIR/env/statedetails.xml* file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Perform the following steps to change the Status Window Table default icons.

The icons contained in the package are used for other Save Manager components such as action buttons. Ensure you modify the correct icon files. The icon size should be** 25x25 pixels **to display correctly.

1. Uncompress the *Cat2tciccommon.jar* file by using the `jar xvf Cat2tciccommon.jar` command. Note a JDK toolkit is necessary to access the jar command.

2. Locate the default Status Window Table icons within the *com.ebsolutions.savemanager.modules.images* package. The icon files are in .png .gif and .jpg formats.

3. Replace the existing icons as needed. For example, replace the *product.png* file with a new icon file to replace the current default CATProduct icon.

4. When finished, re-jar the *Cat2tciccommon.jar* file.

5. Place the shell into the root directory of the original file.

6. Enter the `jar cvf Cat2tciccommon.jar com WaitForSemaph.class` command to generate a new jar file containing new icons. The Status Window Table displays new icons.

6.15.1.3 Status Window Table Customization: Column Widths and Titles

Modify the $TCICV5_DIR/env/statedetails.xml file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.15.1.3.1 Adjust Column Width

Adjust the width of the Status Window Table columns by modifying the columns' width attribute found within the `nameRef = State Details Module` attribute. The value of the **column width attribute** specifies the default width of the column in pixels. If no value is displayed next to the column element, the column widths automatically adjust to fit its content.

The width attribute below indicates that the CATIA Status column displays with a column width of 60 pixels.

```
<Column name="catia_status" width="60"/>
```

6.15.1.3.2 Modify Column Titles

Modify column titles of the Status Window Table by modifying the display attribute found within the `nameRef = State Details Module` attribute. The column titles display the value in its `display` attribute. If no value is assigned, the default localized string is used to display the title of the column depending on the `name` attribute value. The display attribute overrides the column default name, but no localization is applied to the display attribute value.

The display attribute value below indicates the CATIA Status column displays *State In CATIA* as its title.

```
<Column name="catia_status" display="State In CATIA"/>
```

6.15.1.4 Status Window Table Customization: Reorder Columns

Modify the *[$TCICV5_DIR/env/statedetails.xml](#)* file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Reorder columns within the Status Window Table by modifying the order of the **column** elements found within `nameRef = State Details Module` attribute.

The columns within the Status Window Table display in the **same** order as displayed within the list. Rearrange the order of the list to rearrange the order of the columns within the table. In the example .xml file shown below, columns in the Status Window Table display in the following order.

```
<Column name="TC Item item_id"/>
<Column name="TC ItemRev item_revision_id"/>
<Column name="document_type"/>
<Column name="CATIA partNumber"/>
<Column name="TC Dst checked_out" />
<Column name="TC Bvr checked_out" />
<Column name="catia_status" width="60"/>
<Column name="tc_status"/>
<Column name="env"/>
```

```
<Column name="is_env_compatible"/>
<Column name="is_latest_rev"/>
```

1. Item ID
2. ItemRev Revision
3. Type
4. CATIA partNumber
5. Dst Checked-Out
6. Bvr Checked-Out
7. CATIA Status
8. Modified In Tc

Use the *statedetails.xml* file to configure columns:

```
    <Column name="TC Item item_id"/>
    <Column name="TC ItemRev item_revision_id"/>
    <Column name="former_file_name"/>
    <Column name="document_type"/>
    <Column name="CATIA partNumber"/>
    <Column name="TC Dst checked_out" />
    <Column name="TC Bvr checked_out" />
    <Column name="TC Dst checked_out_user" />
    <Column name="TC Bvr checked_out_user" />
    <Column name="TC Dst is_modifiable"/>
    <Column name="TC Bvr is_modifiable"/>
    <Column name="TC ItemRev release_status_list"/>
    <Column name="catia_status" width="60"/>
    <Column name="tc_status"/>
    <Column name="env"/>
    <Column name="is_env_compatible"/>
    <Column name="is_latest_rev"/>
    <Column name="modifiable_dataset"/>
    <Column name="modifiable_bvr"/>
```

By default, columns are displayed in the following order:

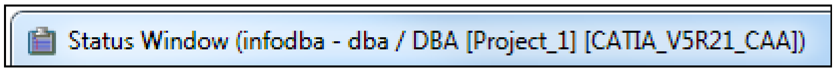
1. Item ID
2. ItemRev Revision
3. File Name

4. Type
5. CATIA partNumber
6. Dst Checked-Out
7. Bvr Checked-Out
8. Dst Checked-Out By
9. Bvr Checked-Out By
10. Dst Is Modifiable
11. Bvr Is Modifiable
12. ItemRev Release Status
13. CATIA Status
14. Modified In Tc
15. CATIA Environment
16. Compatible CATIA Environment

6.15.2 Status Window Title Area

The Status Window title area can display additional information related to the current Teamcenter connection (User, Group, Role) Current Project Name and CATIA environment. The display format of this information is defined in the CATIA_ui_title_format preference.

When this preference is empty, information related to the current Teamcenter connection (User, Group, Role) and CATIA Environment does not display.


Status Window (infodba - dba / DBA [Project_1] [CATIA_V5R21_CAA])

6.15.3 Status Window Toolbar Customization

Modify the *$TCICV5_DIR/env/statedetails.xml* file under the *SaveManager/Module* element to customize this area of the Status Window.

When the *statedetails.xml* file is modified, ensure the encoding of the modified *statedetails.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

6.15.3.1 Customize Status Window Toolbar Buttons

Modify the toolbar buttons of the Status Window Table changing the **DisplayElement** found within the `nameRef = State Details Module` attribute. Each **DisplayElement** defines a graphical button in the toolbar. Refer to the following table for DisplayElement definitions.

Attribute name	Possible values	Description
`showTitle`	`true/false`	Display or hide the button title. By default, the title is not displayed, however, it can be viewed within the button's tooltip.
`nameRef`	`any String`	This attribute overrides the default button title. The value of this attribute displays in the button's tooltip if the showTitle attribute value is **false**. The value of this attribute displays under the icon button if the showTitle attribute value is **true**.
`identifier`	`any String`	This attribute identifies the localization of the button. If the nameRef attribute is declared, the identifier is not used because nameRef will be used "as it is " and no localization proceeds.

Attribute name	Possible values	Description
`image`	`any String`	This attribute contains the pathname of the button's icon. This path can be a package relative path as by default in the .xml file, or an absolute local path. Compatible icons formats are .gif, .jpg and .png.
`class`	`any String`	This attribute is designed for a developer customization purposes and defines the class used to render the button. .
`action`	`any String`	This attribute is designed for a developer customization purposes and defines the action class used when the button is activated.

Every button position can be changed in every toolbar by changing the order of the `DisplayElement` elements in the corresponding toolbar definition.

Within the Integration, another button style is provided to illustrate extending the Status Window components. For example, within the `DisplayElement` element of the right toolbar definition replace the

```
 <class=="com.ebsolutions.savemanager.m
odules.
          SaveModuleToolBarButton"/>
```

definition by

```
<class="com.ebsolutions.savemanager.mo
dules.
       SaveModuleSwingToolBarButton"/>
```

to create a new button style. The button will swing each time the mouse hovers over it.

6.15.3.2 Modify Toolbar Order

Modify the order of the Status Window Table toolbar by reordering the **toolbar elements** found within the `nameRef = State Details Module` attribute. A list of **toolbar elements** defines the Status

Window right vertical toolbars. A graphical separator is automatically added between each toolbar element. Reorder the list of toolbar elements within the .xml file, as needed, to reorder the right side toolbar display.

Use caution when reordering the toolbar elements. Ensure to Copy/Paste the entire toolbar definition block, including the `DisplayElement` elements of each toolbar containing the toolbar button definition. Failure to Copy/Paste the entire information may corrupt the .xml file and prevent the Status Window from being displayed.

6.15.4 Preferences Related to the Status Window

Check-in/Check-out Panel

Display Additional Information Within the Title of the Status Window Panel

Teamcenter must be restarted after modifying preferences.

6.15.4.1 Check-in/Check-out Panel

Set **CATIA_cico_ask_confirmation_action_from_ui** to **true** to display the Check-in / Check-out panel containing the **OK** button and require user interaction to start the action. Set it to **false** to prevent the display of Check-in / Check-out panel and start the action without user interaction. The Check-in/Check-out panel always displays. even when this preference is set to **false** if errors occur.

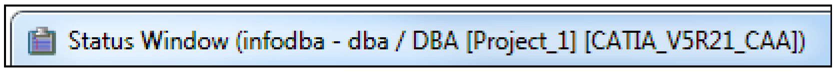
```
CATIA_cico_ask_confirmation_action_from_ui = true
```

The default value is **true**.

6.15.4.2 Display Additional Information Within the Title of the Status Window Panel

The Status Window title area can display additional information related to the current Teamcenter connection (User, Group, Role) Current Project Name and CATIA environment. The display format of this information is defined in the CATIA_ui_title_format preference.

When this preference is empty, information related to the current Teamcenter connection (User, Group, Role) and CATIA Environment does not display.


Status Window (infodba - dba / DBA [Project_1] [CATIA_V5R21_CAA])

{0} = user

{1} = group

{2} = role

{3} = name of the current CATIA environment

{4} = current project name

```
CATIA_ui_title_format = ({0}-{1}/{2} [{4}] [{3}])
```

The default value is **({0} – {1} / {2} [{4}] [{3}])** which displays as *user-group / role [current project name] [name of current CATIA environment]*.

This preference is used for the title of the Save Manager panel, Spreadsheet Editor panel and the State Details panel.

6.16 Teamcenter Loader

Modify the *$TCICV5_DIR/env/tcicloader.xml* file under the loader/module element to customize this area of the Teamcenter Loader panel.

When the *tcicloader.xml* file is modified, ensure the encoding of the modified *savemanager.xml* file is compatible with the printable character set of the .xml file to avoid unpredictable behavior.

Commands, Contextual Menus (in the DisplayElement menu field or in Command name field), Search results (in the History identifier field), can be configured with the *tcicloader.xml.*

When Search has been launched, the query result is saved in *tcicloader.xml* and the search is kept for the next time Teamcenter loader is displays.

```
 <!-- LOADER search History -->
<History identifier="search_results">
<Item uid="Q_YZZFbX4KsTnB"/>
```

```
<Item uid="QyaZZFbX4KsTnB"/>
<Item uid="A2RZZFbX4KsTnB"/>
</History>
```

6.16.1 Hide Commands

Customize the *tcicloader.xml* file to hide some commands in the contextual menu with the Command name field.

```
<Command name="paste"
image="/com/ebsolutions/uimanager/modules/images/loader/paste_16.pn
g"/>    <Command name="create_folder"/>
<Command name="remove_folder"/>
<Command name="load"
image="/com/ebsolutions/uimanager/modules/images/loader/export_32.p
ng"/>    <Command name="open_dataset"
image="/com/ebsolutions/uimanager/modules/images/loader/open_16.png
"/>
<Command name="load_compare"
image="/com/ebsolutions/uimanager/modules/images/loader/export_comp
_16.png"/>
<Command name="load_consulting"
image="/com/ebsolutions/uimanager/modules/images/loader/export_cons
ult_16.png"/>
<Command name="lus"
image="/com/ebsolutions/uimanager/modules/images/loader/lus_16.png"
/>
<Command name="create_exp_spreadsheet"
image="/com/ebsolutions/uimanager/modules/images/loader/create_spre
adsheet_16.png"/>
<Command name="export"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_16.png"/>
<Command name="load_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/export_ref_
16.png"/>
<Command name="export_as_ref"
image="/com/ebsolutions/uimanager/modules/images/loader/file_export
_ref_16.png"/>
<Command name="display_properties"
image="/com/ebsolutions/uimanager/modules/images/loader/properties_
32.png"/>
<Command name="display_structure"
image="/com/ebsolutions/uimanager/modules/images/loader/bomview_32.
png"/>
<Command name="load_selection"/>
<Command name="highlight"
image="/com/ebsolutions/uimanager/modules/images/loader/highlight_s
elected_16.png"/>
<Command name="show_in_teamcenter"
image="/com/ebsolutions/uimanager/modules/images/loader/sendto_16.p
```

```
ng"/>
<Command name="bmml"
image="/com/ebsolutions/uimanager/modules/images/loader/bmml_16.png
"/>
<Command name="save_assy_config"
image="/com/ebsolutions/uimanager/modules/images/save_assy_config_i
con.png"/>
<Command name="close_tab"
image="/com/ebsolutions/uimanager/modules/images/loader/icon_remove
.png"/>
<Command name="close_other_tabs"/>
<Command name="close_all_tabs"
image="/com/ebsolutions/uimanager/modules/images/loader/icon_db_ter
minate_all.png"/>
<Frame name="preview" visible="true" display="false" sizey="240"/>
<Frame name="search_bomlines">
<Column name="and_or"/>
<Column name="prop_name"/>
<Column name="operator"/>
<Column name="value"/>
</Frame>
```

Customize the *tcicloader.xml* file to hide some commands in the CATIA Toolbar using the Command name field.

```
<DisplayElement
         showTitle="false"
         identifier="return_to_catia"
         image="/com/ebsolutions/uimanager/modules/images/loader/re
turn_to_catia_22.png"
         class="com.ebsolutions.loader.ui.LoaderModuleToolBarButton
"
         action="com.ebsolutions.loader.actions.raclessactions.Retu
rnToCATIA"/>
     <DisplayElement
         showTitle="false"
         identifier="start_tc"
         image="/com/ebsolutions/uimanager/modules/images/loader/de
faultapplication_32.png"
         class="com.ebsolutions.loader.ui.LoaderModuleToolBarButton
"
         action="com.ebsolutions.loader.actions.raclessactions.Star
tTeamcenter"/>
```

6.16.2 Impact Analysis

Use the `width` and `position` attributes of Impact Analysis field to customize the display of each area in the Impact Analysis tab of the loader. Each area can by customized and configured using the *tcicloader.xml* file

Use the position attribute to define table position in the tab. The first digit corresponds to the line number in which the table is positioned and the second digit corresponds to the column number in which the table is positioned. Possible values for second digit are 1 or 2.

Array Name	Corresponding Display Name
item revisions	Available Revision
parents	Parents
3d_links	Pointed by/Pointing CATPart, CATProduct, Design table
2d_links	Pointed by/Pointing CATDrawing, CATProcess, CATAnalysis
other_links	Other pointed/pointing documents

Only the item revision field is required.

```
<Frame name="impact_analysis">
<Table name=”itemrevision” position=”1.1”>
<Column name="TC object_string" />
<Column name="type" width=”45” />
<Column name="TC last_mod_date" />
<Column name="TC owning_user" />
<Column name="is_latest_rev" width=”45” />
</Table>
<Table name=”parents” position=”2.1”>
<Column name="TC object_string" />
<Column name="type" width=”45” />
<Column name="TC last_mod_date" />
<Column name="TC owning_user" />
</Table>
<Table name=”3d_links” position=”3.1”>
<Column name="TC object_string" />
<Column name="type" width=”45” />
<Column name="TC last_mod_date" />
<Column name="TC owning_user" />
```

```
<Column name="link_type" />
<Column name="source_target" width=”45” />
</Table>
<Table name=”2d_links” position=”4.1”>
<Column name="TC object_string" />
<Column name="type" width=”45” />
<Column name="TC last_mod_date" />
<Column name="TC owning_user" />
<Column name="link_type" />
<Column name="source_target" width=”45” />
</Table>
<Table name=”other_links” position=”5.1”>
<Column name="TC object_string" />
<Column name="type" width=”45” />
<Column name="TC last_mod_date" />
<Column name="TC owning_user" />
<Column name="link_type" />
<Column name="source_target" width=”45” />
</Table>
</Frame>
```

6.16.3 Loader Size

Use the `sizex` and `sizey` attributes to customize the size of the loader.

```
UIApplication class="com.ebsolutions.loader.ui.LoaderApplication"
            nameRef="Teamcenter Loader"
            image="/com/ebsolutions/uimanager/modules/images/loader
/loader_icon.png"
            sizex="420" sizey="810" lang="">
```

6.16.4 Show in Teamcenter Menu

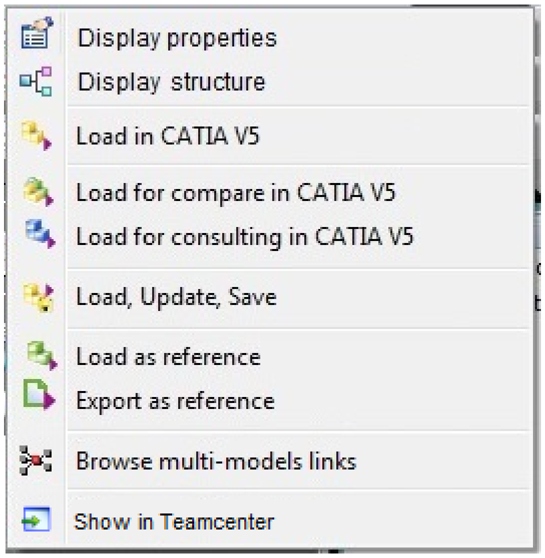
Configure the *tcicloader.xml* file to add or hide the Show in Teamcenter command in the contextual menus of item, item revision and BOM Line selections.

This command is **only** available in the contextual menu for the following installations:

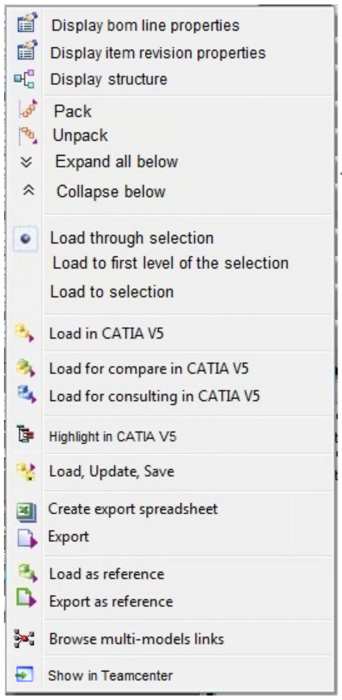
- Teamcenter Loader and RAC
- Teamcenter Loader and Active Workspace
- Teamcenter Loader and Active Workspace and RAC

When Teamcenter loader is connected to the RACLess application and Active Workspace is defined as a client in *client_env* file, use CATIA_show_in_teamcenter_option preference to determine in which client Show in Teamcenter command should be launched between Teamcenter Loader (Loader) or Active Workspace (AW).

On Item/Item Revisions



On BOM Lines



6.16.5 Viewer Size

Use the `sizex` and `sizey` attributes of the Viewer field to customize the size of the JT Viewer in the loader.

```
<Frame name="structure">
<Viewer table_sizex="420"/>
</Frame>
```

7.0 Integration Entry Points

Configuring entry points provides the capability to implement functions called from inside the Integration.

7.1 ITK Functions

7.1.1 Overview

A special ITK library is provided to allow customization of some functions used by Teamcenter Integration for CATIA V5 and modify the way in which information is prepared and saved in Teamcenter.

This library is located in the following directories:

- *%TC_ROOT%\lib\libcatia_custom.lib* for Windows.
- *$TC_ROOT/lib/libcatia_custom.so* for UNIX.

7.1.1.1 Common Header

The **common header** for all ITK functions is as follows:

- `#include <cat_USER_customize.h>`
- `#include <workspaceobject.h>`
- `#include <item.h>`

7.1.1.2 Current Process Identifier

Use the **SaveScriptUsed** global variable, defined in the *cat_USER_customize.h* file, to identify which Save process is the current process in the cat_USER_customize.c and cat_USER_bom_customize.c files. Refer to the table below for values.

This table is filled in by libcatia and is sent to **libcatia_custom**. It can not be modified.

Constant Name	Constant Value	The current process is...
NOT_SAVE_SCRIPT	-1	not a Save process.
SAVE_SCRIPT	0	Save As or it is the Save phase of a Load, Update, Save process.
IMPORT_SCRIPT	1	Import
undefined	2	Save
undefined	3	Save As, for at least one element selected in CATIA.
undefined	4	Create the ImportSpreadsheet
undefined	5	Save Selected (Save Selected Level)
SAVE_TARGET_SCRIPT	6	Save Selected (Save Target)

7.1.1.3 libcatia_tc_custom

The **libcatia_tc_custom** is delivered in sample **dir** of the installation. The `libcatia_tc_custom.c` and `libcatia_tc_custom.h` files are located in this folder.

7.1.1.4 Working Environment

The **EnvTab** global variable, defined in the *cat_USER_customize.h* file, contains working environment table. This table can be used in the cat_USER_customize.c and cat_USER_bom_customize.c files. Each line corresponds to data and indexes are defined in the cat_USER_customize.h file as shown below.

This table is filled in by libcatia and is sent to **libcatia_custom**. It can not be modified.

Constant Name	Constant Value	Corresponding Variable or Preference Value
CAT_TMP_DIR	0	Value of <TcIC_tmp_dir> in the tcic_var_env.xml file.
CAT_FILES_DIR	1	Value of <TcIC_files_dir> in the tcic_var_env.xml file.
IM_ITEM_TYPE	2	CATIA_item_type
PART_BOUNDING_BOX_CALCULATION	3	CATIA_part_bounding_box_calculation_2
CHAR_SEP_ITEMID_ITEMREV	4	FLColumnCatIVFSeparatorPref (Teamcenter preference)
REFERENCE_NAME_SEPARATOR	5	CATIA_partnumber_separator

Constant Name	Constant Value	Corresponding Variable or Preference Value
CATIAV5_SEED_PART	6	CATIA_seed_part_id
CATIAV5_SEED_PRODUCT	7	CATIA_seed_product_id
CATIAV5_SEED_DRAWING	8	CATIA_seed_drawing_id
IM_CATIA_REFERENCE_PRODUCT	9	CATIA_reference_product_partnumber
LANGUAGE	10	Language of the client side
---	11-12	*Reserved for internal use only*
IFS_SPRDSHEET_EXP_PATH	13	Value of <TcIC_export_spreadsheet_path> in the tcic_var_env.xml file
IFS_SPRDSHEET_IMP_PATH	14	Value of <TcIC_import_spreadsheet_path> in the tcic_var_env.xml file
CATIAV5_SEED_PROCESS	15	CATIA_seed_process_id
CATIAV5_SEED_ANALYSIS	16	CATIA_seed_process_id

Constant Name	Constant Value	Corresponding Variable or Preference Value
---	18	*Reserved for internal use only*
CteEXPORT_DLNAME_REQUIRED	19	CATIA_DLName_required_on_export

7.1.2 ItemID/Item Revision

The `ItemID` and the `ItemRevision` are controlled by the standard Teamcenter ITK functions `USER_new_item_id` and `USER_new_revision_id`.

In the PSE window, the sequence number is controlled by the `USER_ask_for_new_sequence_no` ITK function. Refer to the Teamcenter documentation for specific information on these three functions.

7.1.3 List of ITK Functions

7.1.3.1 Assembly Configuration

7.1.3.1.1 Dataset

```
extern int ExportedByCatiaCustom
cat_USER_assyconfig_dataset_info
(tag_t i_itemrev_tag, /* <I> */
tag_t i_bomwindow_tag, /* <I> */
tag_t i_dataset_type_tag,  /* <I> */
tag_t i_relation_type_tag, /* <I> */
char  io_dataset_id[AE_dataset_id_size_c+1], /* <IO> */
char  io_dataset_rev [AE_dataset_rev_size_c+1],  /* <IO> */
char  io_dataset_name[WSO_name_size_c+1],   /* <IO> */
char  io_dataset_desc [WSO_desc_size_c+1])  /* <IO> */
```

Attributes

`i_itemrev_tag` : tag of the item revision top line where the Assembly Configuration dataset will be attached;
`i_bomwindow_tag`: tag of the BOM window
`i_dataset_type_tag` : the tag of the Assembly Configuration dataset type
`i_relation_type_tag`: the tag of the relation type between the item revision and the Assembly Configuration dataset
`io_dataset_id` : the dataset id built by the function
`io_dataset_rev`: the dataset rev built by the function
`io_dataset_name`: the dataset name built by the function
`io_dataset_desc`: the dataset description built by the function

Description

This function is used before an Assembly Configuration dataset is created. It returns the id, the revision, the name and description with which the dataset will be created, according to information retrieved from the item revision, dataset type and relation.

7.1.3.1.2 XML Filename

```
extern int ExportedByCatiaCustom
cat_USER_build_assyconfig_file_name
tag_t i_itemrev_tag, /* <I> */
tag_t i_bomwindow_tag, /* <I> */
tag_t i_dataset_type_tag,  /* <I> */
char **iop_file_name ) /* <IO> */
```

Attributes

`i_itemrev_tag` : tag of the item revision top line where the Assembly Configuration dataset will be attached;
`i_bomwindow_tag`: tag of the BOM window
`i_dataset_type_tag` : the tag of the Assembly Configuration dataset type
`iop_file_name`: the filename of the assembly configuration xml file built by the function.

Description

This function allows the user to define the Assembly Configuration name to save in the dataset. By default, this function returns NULL. In this case, the file name is built by the Integration as following :

*<itemId>_<itemRevId>_<RevRule> **

*RevRule corresponds to the active revision rule name of the BOM window.

7.1.3.2 Auxiliary Files

cat_USER_auxiliaryFilesInfo customizes the filename of auxiliary files. This function is used during save processes.

```
extern int ExportedByCatiaCustom
cat_USER_auxiliaryFilesInfo (
     StructUserDerivedFiles_s i_primary_element,    /* <I> */
            int i_auxiliaryfiles_tab_size,    /* <I> */
            StructUserDerivedFiles_s *iop_auxiliaryfiles_tab) /*
<I/O> */
{
return(!ITK_ok);
}

typedef struct StructUserDerivedFiles {
tag_t itemrev_tag;               /* <I> item rev tag */
tag_t dataset_tag;               /* <I> dataset tag */
logical created_in_tc           /* <I> flag used to indicate if
the dataset has been created during the current process */
char *original_file_name;     /* <I> name of the file on disk (name
of external files, null if auxiliary files)*/
char *file_name;              /* <I/O> name of the file built by
the Integration: customizable in the entry point */}
StructUserDerivedFiles_s;
```

Attributes

`i_primary_element` : information about the primary dataset (StructUserDerivedFiles structure);
`i_auxiliaryfiles_tab_size`: size of `iop_auxiliaryfiles_tab` tab;
`iop_auxiliaryfiles_tab` : tab of auxiliary files related to the primary datatset

Description

This method is used to customize the filename of Auxiliary files. This function is used during the save and save as/revise post-action processes. The default value for the filename is **datasetname_datasetuuid**.

The original_file_name parameter must be copied into the file_name parameter of the StructUserDerivedFiles structure to retain the CATIA original file name.

7.1.3.3 BOM Line Validation/CATPart, CATProduct Datasets

7.1.3.3.1 cat_USER_verifyDatasetCreation

cat_USER_verifyDatasetCreation validates the selected BOM line prior to creating a CATPart or CATProduct dataset.

```
extern int ExportedByCatiaCustom
cat_USER_verifyDatasetCreation (
tag_t          bomline,        /* <I> */
const char*    dsType ,        /* <I> */
logical*       skipCreation,   /* <O> */
char**         message )       /* <O> */
```

Attributes

`bomline` : tag of the selected BOM line;
`dsType` : dataset type corresponding to the selected line;
`skipCreation` : skip dataset creation;
`message` : message to return;

Description

This user exit is called after a preliminary verification of the BOM line is made for dataset creation from the CATIAV5 menu. When the returned error value is not *ITK_ok* the processing of the structure is aborted. The message returned is reported as an error. When the returned error value is *ITK_ok*, and the skipCreation value is set to `true`, the BOM line is skipped and the message is reported to the user as a warning.

7.1.3.4 Check In/Check Out

7.1.3.4.1 cat_USER_entry_point_after_check_all_components

```
extern int ExportedByCatiaCustom
cat_USER_entry_point_after_check_all_components (
int          check_in_out,    /* <I>
*/                                                     USER_Part_s
*datasets_tab,    /* <I>
*/                                                     int          n
b_datasets )    /* <I>*/
```

Attributes

`check_in_out` : value of the check command selected in CATIA: 0 if check out; 1 if check in;
`datasets_tab` : tab of checked datasets;
`nb_datasets` : number of checked datasets.

Description

This function defines an entry point after the Check-In or Check-Out command in Teamcenter.

If the command is Check-in, the USER_Part_s tab contains all components checked in by the process.

- If a component is already checked in before the check in process, this component will not be set in the tab.

If the command is Check-out, the USER_Part_s tab contains all components checked out by the process.

- If a component is already checked out before the check out process, this component will not be set in the tab.

If the command is Save, the USER_Part_s tab contains all components checked out or checked in by the process.

- If a component is checked out and the Save command is performed with the Check-in option, the component is put in the tab.

- If a component is Save as New, but without the Check-in option, the component is put in the tab. Otherwise, the component is not put in the tab.

The USER_Part_s type is a structure defined as follows :

```
typedef struct USER_Part
{
      char dataset_uid[28];
      char item_id [ITEM_id_size_c];
      char item_rev_id[ITEM_id_size_c];
} USER_Part_s;
```

7.1.3.5 Dataset

7.1.3.5.1 cat_USER_build_dataset_name

```
extern int ExportedByCatiaCustom

cat_USER_build_dataset_name (tag_t i_item_rev, /* <I>
*/
                            tag_t i_dataset_type,  /* <I>
*/
                            tag_t i_relation_type,  /* <I> */
                            char
io_dataset_name[WSO_name_size_c+1]) /* <O> */
```

Attributes

`i_item_rev` : The tag of the Item Revision;
`i_dataset_type` : The tag of the dataset type
`i_relation_type` : The tag the relation type between the dataset and the item revision
`io_dataset_name` : The dataset name built by the function

Description

This function is used before a Dataset is created from a CATIA V5 component. It returns the Dataset name.

By default, the name of the dataset is the result of the call to USER_new_dataset_name or as entered in the Save dialog.

The USER_new_dataset_name ITK function does not take the dataset name into account.
When USER_new_dataset_name returns an error or a null dataset_name, the cat_USER_build_dataset_name also returns an error or an empty dataset_name. When the dataset name is empty, the dataset name is built with the ItemId/ItemRevId rule.
If errors occur, an Unable to create the dataset name error message displays and the Save process stops.

7.1.3.5.2 cat_USER_buildFileName

```
extern int ExportedByCatiaCustom
cat_USER_buildFileName ( tag_t    i_item_rev_tag,       /* <I> */
                                tag_t    i_dataset_tag,        /*
<I> */
                                char     **iiop_file_name);  /*
<I/OF> */
```

Attributes

`i_item_rev_tag` : tag of the Item Revision
`i_dataset_tag` : tag of the Dataset
`iop_file_name` : the filename built by the function.

Description

This function is used when the Integration builds the FileName's primary object during Save or Load process. It returns the FileName customization implemented in the entry point function.

By default, the FileName of the primary object is build according to CATIA_xxx_file_name_format preference value. For instance, the CATPart documents' FileName is built according to CATIA_part_file_name_format preference value which is itemID_itemRevID, by default.

7.1.3.5.3 cat_USER_buildPartNumber

```
extern int ExportedByCatiaCustom
cat_USER_buildPartNumber ( tag_t    i_item_rev_tag,       /* <I> */
                                tag_t    i_dataset_tag,        /*
<I> */
                                char     **iiop_partnumber);  /*
<I/OF> */
```

Attributes

`i_item_rev_tag` : tag of the Item Revision
`i_dataset_tag` : tag of the Dataset
`iop_partnumber` : the part number built by the function.

Description

This function is used when the Integration builds the PartNumber’s primary object during Save or Load process. It returns the PartNumber customization implemented in the entry point function.

By default, the FileName of the primary object is build according to Teamcenter mapping or the value of the CATIA_MAP_Property_PartNumber preference.

7.1.3.5.4 cat_USER_get_dataset_revision

```
extern int ExportedByCatiaCustom
cat_USER_get_dataset_revision ( tag_t    item_rev_tag,       /* <I>
*/
                                tag_t    dataset_tag,        /* <I>
*/
                                int     *dataset_revision )  /* <O>
*/
```

Attributes

`item_rev_tag` : tag of the Item Revision
`dataset_tag` : tag of the Dataset
`dataset_revision` : returned dataset revision value.

Description

This function defines an entry point to allow the user to define the dataset revision used when creating the dataset, for Save As New and Save As New Revision processes. It is called after creating the dataset.

If this function returns a FailCode or if the dataset revision value is not greater than 1, no dataset revision will be set to the dataset, then the dataset revision will be 1.

If this function returns a valid revision number, and only in this case, this revision number is assigned to the dataset and its anchor. Two saves are performed, one on the dataset anchor and one on the dataset itself.
If this function is not customized or returns a FailCode or a revision number greater than 1, only the dataset will be saved.

7.1.3.5.5 cat_USER_is_default_dataset

```
extern ExportedByCatiaCustom cat_USER_is_default_dataset
             (tag_t       i_item_rev_tag,     /* <I> */
             tag_t      *ip_dataset_tag_tab, /* <I> */
             tag_t      *ip_relation_tag_tab, /* <I> */
             int         i_dst_tab_size,      /* <I> */
             logical    *op_is_default_dataset, /* <O> */
             tag_t      *op_dst_tag) /* <O> */ )
```

Attributes

`i_item_rev_tag` : tag of the item revision;
`ip_dataset_tag_tab` : array of dataset tags, all datasets have the same type (entry point is called several times when different dataset types exist under the item revision)
`ip_relation_tag_tab:` array of relation tags between an item revision and each dataset of i_dataset_tag_tab
`i_dst_tab_size`: size of i_dataset_tag_tab and i_relation_tag_tab
`op_is_default_dataset` : flag indicating if a default dataset is found
`op_dst_tag`: tag of the default dataset: should be not NULLTAG if o_is_default_dataset = true and if there are many datasets of the same type under the item revision (i_dst_tab_size > 1)

Description

This entry point is called during the Load process, when datasets attached to a part item revision are read. It permits UserA to load an assembly with specific foreign files and UserB to load the same assembly with CATPart files depending on settings within Teamcenter preferences.

This function allows the user to indicate which dataset will be a default dataset, i.e., which dataset to load, depending on the dataset type defined in `ip_dataset_tag_tab` list.

The **CATIA_component_dataset_types** setting in Teamcenter preferences is read to define the datasets types' priority. This preference is taken into account when `op_is_default_dataset` parameter returns false.

This function is called for each dataset type defined in the list returned by the **CATIA_CATPart_dataset_type**, **CATIA_catia_dataset_type** and **CATIA_component_dataset_types** preferences on a leaf assembly.

When `op_is_default_dataset` parameter returns true:

- When `i_dst_tab_size` is equal to 1, the default dataset is returned in `i_dataset_tag_tab`.
- When `i_dst_tab_size` is greater than 1, the default dataset is returned in `o_dst_tag`.
- When `o_dst_tag` is NULLTAG, the default dataset search continues.

7.1.3.5.6 cat_USER_validateCacheDatasetForLoad

```
extern int ExportedByCatiaCustom
cat_USER_validateCacheDatasetForLoad(tag_t
i_geo_dataset,                        /* <I> */
                                                        tag_t   i_c
ache_dataset,                    /* <I>
*/                                                         tag_t
i_load_as_cgr_process,            /* <I>
*/                                                         tag_t  *
op_is_cache_dataset_valid)    /* <O> */
```

Attributes

`i_geo_dataset` : tag of the geometry dataset;
`i_cache_dataset` : tag of the cache dataset;
`i_load_as_cgr_process` : boolean to specify when it is a *Load as cgr* process;
`op_is_cache_dataset_valid`: boolean to specify if the cache dataset is empty

Description

This entry point is called during the Load process, when datasets attached to a part item revision are read. It allows the user to customize the criteria used to determine when a cache dataset is valid for a **Load as cgr** process. When a cache dataset is not valid, the geometry dataset is loaded instead of the cache, even when the **Load as cgr** option is activated.

This entry point reads the CATIA_cache_validate_dataset_for_load preference but takes priority over the preference value.

7.1.3.6 Export as reference and Load as reference

This method enhances the **Export as reference** and **Load as reference** processes which permits exporting and loading structures to a reference directory.

7.1.3.6.1 cat_USER_buildReferencePrefixSuffix

```
cat_USER_buildReferencePrefixSuffix(tag_t
i_bom_window,tag_t  i_item_rev_tag,
             tag_t i_dataset_tag,
              char  *i_named_ref_type,
              tag_t  i_file_tag,
              EnumUSERReferenceType i_type,
              char **op_prefix,
                              char **op_suffix)
```

Attributes

`i_bom_window`: BOM window tag;
`i_item_rev_tag:` Item revision tag;
`i_dataset_tag`: Dataset tag;
`i_named_ref_type`: Named reference type to be loaded;
`i_file_tag`: file tag in the dataset's named reference;
`i_type`: data type (must be a top level, component node, product node or leaf);
`op_prefix`: prefix;
`op_suffix`: suffix;

Description

This method enables the user to define an additional prefix and suffix for the file name in order to avoid file conflicts and staging conflicts. By default, it returns !*ITK_ok*, therefore only the **CATIA_reference_prefix** preference value is applied.

The file names are built according to the preference, prefix and suffix values. Refer to the following table:

Prefix Value?	Suffix Value?	Result
no	no	file name = valuePref_fileName

Prefix Value?	Suffix Value?	Result
yes	no	file name = valuePref_FonctionPrefixValue_fileName
no	yes	file name = valuePref_fileName_FonctionSuffixValue
yes	yes	file name = aluePref_FonctionPrefixValue_fileName_FonctionSuffixValue

7.1.3.6.2 cat_USER_buildReferencePrefixSuffixPartNumber

```
cat_USER_buildReferencePrefixSuffixPartNumber(tag_t
i_bom_window,tag_t  i_item_rev_tag,
             tag_t i_dataset_tag,
              char  *i_named_ref_type,
              tag_t  i_file_tag,
              EnumUSERReferenceType i_type,
              char **op_prefix,
                              char **op_suffix)
```

Attributes

`i_bom_window`: BOM window tag;
`i_item_rev_tag`: Item revision tag;
`i_dataset_tag`: Dataset tag;
`i_named_ref_type`: Named reference type to be loaded;
`i_file_tag`: file tag in the dataset's named reference;
`i_type`: data type (must be a top level, component node, product node or leaf);
`op_prefix`: prefix;
`op_suffix`: suffix;

Description

This method enables the user to define an additional prefix and suffix for the part in order to avoid part number conflicts. By default, it returns !*ITK_ok*, therefore only the **CATIA_reference_prefix_partnumber** preference value is applied.

The part numbers are built according to the preference, prefix and suffix values. Refer to the following table:

Prefix Value?	Suffix Value?	Result
no	no	part number = valuePref_partNumber
yes	no	part number = valuePref_FonctionPrefixValue_partNumber
no	yes	part number = valuePref_partNumber_FonctionSuffixValue
yes	yes	part number = valuePref_FonctionPrefixValue_partNumber_FonctionSuffixValue

7.1.3.7 External Files

cat_USER_externalFilesInfo customizes the filename of external files. This function is used during save processes.

```
extern int ExportedByCatiaCustom
cat_USER_externalFilesInfo (
     StructUserDerivedFiles_s i_primary_element,    /* <I> */
            int i_externalfiles_tab_size,    /* <I> */
            StructUserDerivedFiles_s *iop_externalfiles_tab) /*
<I/O> */
{
return(!ITK_ok);
}

typedef struct StructUserDerivedFiles {
tag_t itemrev_tag;                /* <I> item rev tag */
tag_t dataset_tag;                /* <I> dataset tag */
logical created_in_tc            /* <I> flag used to indicate if
the dataset has been created during the current process */
char *original_file_name;     /* <I> name of the file on disk (name
of external files, null if external files)*/
char *file_name;              /* <I/O> name of the file built by
the Integration: customizable in the entry point */}
StructUserDerivedFiles_s;
```

Attributes

`i_primary_element` : information about the primary dataset (StructUserDerivedFiles structure);
`i_externalfiles_tab_size`: size of `iop_externalfiles_tab` tab;
`iop_externalfiles_tab` : tab of external files related to the primary datatset

Description

This method is used to customize the filename of External files. This function is used during save processes. The default value for the filename is **primary document filename.external file extension**.

The original_file_name parameter must be copied into the file_name parameter of the StructUserDerivedFiles structure to retain the CATIA original file name.

7.1.3.8 Foreign Files

7.1.3.8.1 cat_USER_build_foreign_file_name

```
extern ExportedByCatiaCustom
char *cat_USER_build_foreign_file_name ( tag_t ItemRevTag,     /*
<I> */                                          tag_t
DatasetTag,     /* <I>
*/                                          char *old_file_name )
/* <I> */
```

Attributes

ItemRevTag : tag of the item revision where the foreign dataset will be attached;
`DatasetTag` : tag of the foreign dataset where to save the foreign file;
`old_file_name` : old foreign file name.

Description

This function defines an entry point to allow the user to define the foreign file name to save in the dataset. By default, this function returns NULL. In this case, the file name is built like following:

`<old_file_name>_<dataset_uid>.<extension>`

Modifications to the dataset name or the dataset description in this function can be performed. However, do not save the dataset after these modifications using the AOM_save function, it will be performed later in the code.

7.1.3.9 Import Spreadsheet

7.1.3.9.1 cat_USER_entry_point_customize_sprdsheet

```
extern int ExportedByCatiaCustom
cat_USER_entry_point_customize_sprdsheet ( EXP_Part_s *part )  /*
<I/OF> */
```

Attributes

`part` : An EXP_Part_s structure.

Description

This function customizes what is written to the spreadsheet during the Import process. It takes an EXP_Part_s structure as parameter, defined below.

```
typedef struct EXP_Part
{
        tag_t itemRev_tag;
        tag_t *dst_tag;
        int dst_nbr;
        char OEM_fileName[257];
        char OEM_partNbr[129];
        char OEM_def[129];
        char OEM_rev[129];
        char OEM_nomenc[129];
        char OEM_desc[257];
        char OEM_dlname[129];
                char item_id[128];
        char itemRev_id[128];
        char item_type[ITEM_type_size_c + 1];
        char dst_type[WSO_name_size_c + 1];
        char item_co[8];
        char itemRev_co[8];
        char dst_co[8];
        char bvr_co[8];
        char status[8];
        char **userProps;
        int nbUserProps;
        char **newData;
        int newData_nbr;
} EXP_Part_s;
```

The newData field can be used to add data in the spreadsheet. These data will then be added at the end of the current line. The `newData_nbr` must be set to the number of strings added at the end of the line.

7.1.3.10 Item Revision

7.1.3.10.1 cat_USER_find_stencil_item_rev_tag

```
extern int ExportedByCatiaCustom
cat_USER_find_stencil_item_rev_tag ( const char  item_id[32],    /*
<I>
*/                                      tag_t      *item_tag,
 /* <0>
*/                                      tag_t      *item_rev_tag
)  /* <O> */
```

Attributes

`item_id` : The ID of the Item;
`item_tag` : The Item Tag returned by the function;
`item_rev_tag` : The Item Revision Tag returned by the function.

Description

This function is used before a Dataset without a reference name is loaded in CATIA V5. It returns the item tag and the item revision tag of the seed document's item ID. By default, the item revision tag is the latest item revision tag.

7.1.3.11 Item and Item Revision

7.1.3.11.1 cat_USER_verify_item_and_rev

```
extern int ExportedByCatiaCustom

cat_USER_verify_item_and_rev ( char   item_id[32],      /* <I> */
                               char   item_rev_id[32],  /* <I> */
                               char   item_type[32],    /* <I> */
                               tag_t  item_tag,         /* <I> */
                               tag_t  item_rev_tag )    /* <I> */
```

Attributes

`item_id` : ID of the Item;
`item_rev_id` : ID of the Item Revision;
`item_type` : type of the Item;
`item_tag` : tag of the Item;
`item_rev_tag` : tag of the Item Revision.

Description

This function is used to define an entry point during the save and supplier import form processes. It allows the user to validate the IDs (according to naming rules, for instance) and returns modified values under some circumstances.

For the import process, this entry point is used during the reading of the desc file. If one or more IDs are invalid, the process is stopped and an error file is created listing the invalid IDs.

For the save process, this entry point is used after OK is selected in the Save Item panel. If an ID is invalid, the panel is displayed again.

7.1.3.12 Item Revision Rule

7.1.3.12.1 cat_USER_getLatestRevision

cat_USER_getLatestRevision returns the latest item revision of the input revision list corresponding to the **CATIA_latest_revision_set_type** preference definition.

```
extern int ExportedByCatiaCustom
cat_USER_getLatestRevision (tag_t  *ip_item_rev_list ,         /*
<I> */
                           int     i_item_rev_list_size ,     /*
<I> */
```

```
                          tag_t   *op_item_rev_tag)         /*
<O> */
```

Attributes

`ip_item_rev_list`: list of item revision tag;
`i_item_rev_list_size`: size of item revision tag list;
`op_item_rev_tag` : NULLTAG if no match is found or tag of the returned item revision;

Description

This function is used to customize the latest revision type to return and is used in the Import, Replace by revision and Save processes. Use the CATIA_latest_revision_set_type preference to filter elements returned by Import Query during spreadsheet creation and to determine the latest revision type to return.

7.1.3.13 JT

7.1.3.13.1 cat_PV_catia2jt_next

```
extern int ExportedByCatia cat_PV_catia2jt_next ( tag_t
item_rev_tag,   /* <I>
*/                                                  int   DMdtsPro
cess )  /* <I> */
```

Attributes

`item_rev_tag` : tag of the Item Revision;
`DMdtsProcess` : must be set to 1 to avoid a recursivity.

Description

This function creates or updates the JT dataset if necessary to save a new jt file and calls the catia2jt script.

7.1.3.13.2 cat_USER_buildJTFromNXFilename

```
extern int ExportedByCatiaCustom
cat_USER_buildJTFromNXFilename(tag_t  i_item_rev_tag,          /*
<I> */
                              tag_t  i_dataset_tag,            /*
<I> */
                              char   *i_named_ref_type,        /*
<I> */
```

```
                                tag_t   i_file_tag,               /*
<I> */
                                char   *i_foreignfilename_format, /*
<I> */
                                char **iop_exported_file_name);  /*
<I/OF> */
```

Attributes

`i_item_rev_tag`: Item revision tag;
`i_dataset_tag`: Dataset tag of the JT from NX;
`i_named_ref_type`: Named reference type to be loaded;
`i_file_tag`: File tag of the JT from NX;
`i_foreignfilename_format`: CATIA_foreignfile_file_name_format preference value;
`io_exported_file_name`: Filename used to export the JT file from the staging directory;

Description

This method builds a unique filename for JT datasets created with NX. By default, the filename is built based on the CATIA_foreignfile_file_name_format preference value. This function is called in all load processes.

Teamcenter Integration for CATIA V5 supports direct loading of JTs to CATIA V5 when a license to use Theorem JT to load to CATIA V5 is available. JTs are delivered and inserted from CATIA and links to the JT file are maintained, with correct configurations. JT files are loaded to CATIA using Visualization or Design Mode, depending on the CATIA configuration.

When the returned filename is not unique in the assembly to be loaded, a filename conflict displays.

When the function returns a failcode or a null filename, the filename for the Named Reference is used to export the JT file from the staging directory.

7.1.3.14 Load

7.1.3.14.1 cat_USER_hasChildren

```
extern int ExportedByCatiaCustom
cat_USER_haschildren( tag_t       i_bom_line,         /* <I> */
                       tag_t      i_item_rev_tag,     /* <I> */
                       tag_t      i_bvr_tag,          /* <I> */
                       logical    *op_answer);        /* <O> */
```

Attributes

`i_item_rev_tag` : The tag of the Item Revision
`i_bom_line` : The tag of the BOMLine
`i_bvr_tag` : The tag of the BOMViewRev
`op_answer` : TRUE if the item corresponds to a Product.

Description

This function is used to determine if a given item will be loaded in CATIA as a Product or a Part depending on specific conditions.

Five boolean variables are available to define various combinations in order to determine which type of document will be loaded in CATIA.

- `ExistBVR` : Set to **true** if under the Item when a BOMView type is defined in Teamcenter Preferences.
- `EmptyBvr` : Set to **true** if the previous variable, `ExistBVR`, is also **true** and theassociated BOMView Revision does not contain occurrences.
- `ExistCATProduct` : Set to **true** if the dataset type under the Item revision is defined in Teamcenter Preferences.
- `ExistCATPart` : Set to **true** if the dataset type under the Item revision is defined in Teamcenter Preferences.
- `ExistOtherDataset` : Set to **true** if the dataset type under the Item revision is defined in Teamcenter Preferences.

In this function, the **CATIA_customize_load_process** setting defined in Teamcenter Preferences, is used by the `VarCATIA_CUSTOM_LOAD` variable to determine the criterion used for the Load in CATIA.

- When the CATIA_customize_load_process option is set to **0** (default value) in Teamcenter Preferences the process analyzes the branches of the BOM tree, and stops when a node does not contain a valid CATIA BOM. In this case, `cat_USER_hasChildren` returns op_answer=`FALSE` and the item is considered a leaf of the tree (component).
- When CATIA_customize_load_process option is set to **1** in Teamcenter Preferences, the process analyzes the part and product datasets under the given item revision and checks whether the BOMView revision contains occurrences.
- When CATIA_customize_load_process option is set to **2** in Teamcenter Preferences, the process analyzes the part, product, V4 model and foreign file datasets under the given item revision and checks whether the BOMView Revision contains occurrences.

The `cat_USER_hasChildren` function returns op_answer=`FALSE`, indicating a part will be loaded, in three cases:

- A CATPart/V4/Foreign File dataset exists and a CATProduct dataset and BOMView Revision do not exist.
- A CATPart/V4/Foreign File dataset and BOMView Revision containing no occurrences exists and a CATProduct dataset does not.
- A dataset and BOMView Revision do not exist.

7.1.3.15 Occurrences

7.1.3.15.1 cat_USER_isOccurrenceNonCAD

File location: `cat_USER_bom_customize.c`
customer entry point: `cat_USER_isOccurrenceNonCAD`
default return value: `false`

Attributes

`i_bom_line_tag <I>`: tag of the selected BOM line;
`iopp_message_tab <IOF>` : contains customer messages; a *NULL* pointer is given for the first call, customer must allow this array themselves;
`iop_size_message_tab <IO>` : iopp_message_tab array size.

Use Teamcenter allocation functions (MEM_alloc, MEM_crealloc...). Do not use Standard C memory allocation functions.

Description

This entry point is called during the **Load** and **Save** process, to check whether the instance corresponding to the i_bom_line_tag must be loaded.
When this function returns **true** for i_bom_line_tag, the instance is not loaded in CATIA; when it returns **false** (default), the current instance is loaded in CATIA.

Remarks

- The Entry point is not called for the root product of an assembly (top level); when the entry point always returns false, only the root product is loaded.
- An error message displays when the PSE selection is a non-CAD occurrence or when one of its parents is a non-CAD occurrence.

7.1.3.15.2 cat_USER_isOccurrenceToBeLoaded

File location: `cat_USER_bom_customize.c`
Customer Entry Point: `cat_USER_isOccurrenceToBeLoaded`
default return value: `true`

Attributes

`i_top_line_item_rev_tag`, /* <I> tag of top Node (assembly context)*/
`i_occuid_path`: <I> occuid path of current item;
`i_parent_item_rev_tag`: <I> tag of the parent item revision;
`i_parent_bomvr_tag`: <I> tag of the parent BOMview revision;
`i_item_rev_tag`: <I> tag of the item revision;
`i_occ_tag` : <I> tag of the occurrence.

Description

This function is used during Load processes. The cat_USER_isOccurrenceToBeLoaded entry point is called to determine if the instance corresponding to *i_occ_tag* must be loaded. If this function returns `false` for i_occ_tag, the instance is not loaded in CATIA.

The entry point is not called for the assembly root product. If the entry point always returns `false`, only the root product loads.
A warning message, specific to the Load Merge Target, Load Merge Selected, and Refresh processes, displays when items are bypassed by this entry point.

7.1.3.16 Replace by revision

The cat_USER_getItemRevisionForReplaceByRevision method enhances the Replace by revision process which permits revising an item in CATIA with a specific revision.

```
cat_USER_getItemRevisionForReplaceByRevision
                (tag_t i_item_tag,
                 tag_t i_item_rev_tag,
                 tag_t *o_item_rev_list,
                 int *o_item_rev_list_size);
```

Attributes

`i_tag`: Item tag of the selected element to be replaced;
`i_item_rev_tag`: item revision tag of the selected element to be replaced;
`o_item_rev_list`: list of item revisions available for Replace By revision;
`o_item_rev_list_size`: list size;

Description

This method enables the user to define a list of item revisions to be displayed in the Replace By Revision panel.

By default, this method returns all item revisions accessible to the current user except the one corresponding to `i_item_rev_tag`.

7.1.3.17 Save

7.1.3.17.1 cat_USER_entry_point_after_save_all_files

```
extern int ExportedByCatiaCustom
cat_USER_entry_point_after_save_all_files ( USER_Part_s
*datasets_tab,  /* <I>
*/                                              int          nb_data
sets )  /* <I> */
```

Attributes

`datasets_tab` : The tab of saved datasets;
`nb_datasets` : The number of saved datasets.

Description

This function defines an entry point after the save of all files and all multi-links of an assembly during the save process. The USER_Part_s type is a structure defined like following :

```
typedef struct USER_Part
{
      char dataset_uid[28];
      char item_id [ITEM_id_size_c];
      char item_rev_id[ITEM_id_size_c];
} USER_Part_s;
```

For example, use the cat_PV_catia2jt_next function to use the JT translator build JT datasets for all saved parts.

7.1.3.18 Save New Revision

7.1.3.18.1 cat_USER_validateForSaveNewRevision

```
extern int ExportedByCatiaCustom

cat_USER_validateForSaveNewRevision (
tag_t i_item_tag,           /* <I> */
tag_t i_old_item_rev_tag,   /* <I> */
char *i_new_item_rev_id,    /* <I> */
tag_t i_dataset_type,       /* <I> */
logical *op_is_valid,       /* <O> */
char **op_message);    /* <O> */
```

Attributes

`i_item_tag` : item tag;
`i_old_item_rev_tag`: item rev tag of the old revision. The value is NULLTAG, by default, during the Import/Create ImportSpreadsheet processes;
`i_new_item_rev_id` : item revision ID of the new revision;
`i_dataset_type`: dataset type tag;
`op_is_valid`: boolean used to validate the new revision;
`op_message`: message displayed in the Save Manager

Description

This function is used to validate the Save New Revision or Import New Revision mode.

- When `op_is_valid` = true; the default behavior is used.
- When `op_is_valid` = false:
  - The Save process is not available and a message, defined by `op_message` displays in the Save Manager. If `op_message` has not been defined, the default message, *Unable to create a new revision* displays.

7.1.3.19 Secondary Dataset

7.1.3.19.1 cat_USER_secondary_dataset_info

```
extern int ExportedByCatiaCustom
cat_USER_secondary_dataset_info(
tag_t ItemRevTag,                             /* <I> */
tag_t primary_object_tag,                     /* <I> */
tag_t secondary_dataset_type_tag,             /* <I> */
tag_t relation_type_tag,                      /* <I> */
char  DstName[WSO_name_size_c+1],             /* <IO> */
char  DatasetDescription[WSO_desc_size_c+1])  /* <IO>*/
```

Attributes

`ItemRevTag` : tag of the Item Revision;
`primary_object_tag` : tag of the primary object
`secondary_dataset_type_tag` : the tag of the secondary dataset type
`relation_type_tag` : the tag of the relation
`DstName` : the dataset name built by the function. It is truncated after the function call to WSO_name_size_c or TC_LEGACY_ID_NAME_SIZE according to the TC_Allow_Longer_ID_Name Teamcenter Preference
`DatasetDescription` : the description of the dataset

Description

This function is used before a secondary dataset is created. It returns the name and description with which the dataset will be created, according to information retrieved from the Item Revision and the Drawing dataset the secondary dataset will be linked to. This function is currently used for:

- Auxiliary datasets of CATParts, CATProducts, CATProcesses, and CATAnalysis'
- Auxiliary vectorial dataset
- CATComputation and CATAnalysis datasets
- CATCache datasets
- External datasets of CATParts, CATProducts, CATProcesses, CATAnalysis', and CATDrawings

If the auxiliary document is vectorial, the dataset name is defined by Item ID and ItemRev ID, by default.

If the auxiliary document is not vectorial, the dataset name is defined by sheet referenced by the auxiliary dataset, by default.

7.1.3.19.2 cat_USER_secondary_parts_info

```
extern int ExportedByCatiaCustom
cat_USER_secondary_parts_info(
               StructUserSecondaryPart_s io_parent_prod,     /* <I>
*/
               int i_secondary_parts_tab_size,        /* <I> */
               StructUserSecondaryPart_s
**iop_secondary_parts_tab)       /* <I/O> */

typedef struct StructUserSecondaryPart
{
      tag_t dataset_tag;                           /* <I> */
      logical modified_in_catia;                   /* <I> */
      char *original_file_name;                    /* <I> */
      char *original_part_number;                  /* <I> */
      char *file_name;                             /* <IO> */
      char *part_number;                           /* <IO> */
} StructUserSecondaryPart_s;
```

Attributes

`io_parent_prod` : information about the parent product of the secondary part;
`i_secondary_parts_tab_size`: size of iop_secondary_parts_tab;
`iop_secondary_parts_tab` : secondary part default value.

Description

This method is used to customize the filename and PartNumber of Secondary Parts. This function is used during save processes.

The default value for the filename is the CATProduct filename followed by dot (.) then an index. The default value for the partnumber is the CATProduct partnumber followed by dot (.) then the same filename index.

The original_file_name parameter must be copied into the file_name parameter and the original_part_number parameter must be copied into the part_number parameter of the StructUserSecondaryPart structure to retain the CATIA original file name and part number.

7.1.3.20 Title Block

7.1.3.20.1 cat_USER_write_TTB_values

```
extern int ExportedByCatiaCustom
     cat_USER_write_TTB_values ( char ***char_array, /* <OF> */
                                 int *size_array,    /* <O> */
                                 tag_t ItemRevTag,   /* <I> */
                                 tag_t DatasetTag )  /* <I> */
```

Attributes

`char_array` : array to be written in, array transferred to client;
`size_array` : number of lines;
`ItemRevTag` : tag of the Item Revision;
`DatasetTag` : tag of the Dataset.

Description

This function sends the user-defined name/value pairs to CATIA V5 in order to complete the title block.
The pairs are sent via the string array because the file is written by the client.
Each line shall be of the form "<type> <name>|<value>"
By default, <type> can be an Item, ItemRev, Dst, ItemMaster, ItemRevMaster and <name> is an iMAN attributes name, such as Description, Name, Owner, etc.

```
Item Name|GENERAL ASSEMBLY
ItemRev Revision|A
Mytype MY_ATTRIBUTE|MY_CHARACTER_STRING
```

The last example shows a line with a type, a name and a value that are not from Teamcenter, but that can be built in the function cat_USER_write_TTB_values.

This line must be written into the array like following :

```
char **array = (char**)MEM_alloc(sizeof(char*));
array[0]=(char*)MEM_alloc(128*sizeof(char));
strcpy(array[0],"Mytype MY_ATTRIBUTE|MY_CHARACTER_STRING");
(*char_array)=array;
(*size_array)=1;
```

With this example, the identifier of the CATIA V5 text to be updated should be `IM_CATIA_MYTYPE MY_ATTRIBUTE.` Therefore, the CATIA V5 text will be replaced by MY_CHARACTER_STRING. The information can be retrieved from the item revision or the dataset.

7.1.4 TC_Custom Libraries

UNIX
Windows
Sample Scripts

Ensure C compiler is used to compile and that the compiler version matches Teamcenter requirements listed in the Teamcenter documentation.

Use the ITK function files to generate the `libcatia_tc_custom.lib` and `libcatia_tc_custom.dll` files for Windows or the `libcatia_tc_custom.so` or `libcatia_tc_custom.sl` files for UNIX, depending on platform.

7.1.4.1 UNIX

Perform the following steps to generate and use the UNIX **libcatia_tc_custom** shared library.

1. Locate the following three scripts within the *$TCICV5_DIR/sample* directory.
   - compile_libcatia_tc_custom
   - compile
   - link_custom_tc_exits
2. Ensure you have sufficient rights to execute scripts, then edit compile_libcatia_tc_custom, as required (enter the value for %TC_ROOT% corresponding to the Teamcenter installation location).

   Administrator privileges may be required to perform this operation.

3. Read the warning below, then execute the compile_libcatia_tc_custom script.

   The resulting libcatia_tc_custom library is placed under the $TC_ROOT/lib directory. To avoid copying the shared library under the $TC_ROOT/lib directory, edit the link_tc_custom_exits script by uncommenting the mv $customLibrary ${TC_USER_LIB} line at the end of the script

7.1.4.2 Windows

Perform the following steps to generate and use the Windows **libcatia_tc_custom.dll** shared library.

1. Locate the following three batch files within the *%TCICV5_DIR%\sample* directory.
    - compile_libcatia_tc_custom.bat
    - compile.bat
    - link_tc_custom_exits.bat
2. Set VSINSTALLDIR=C:\Program Files\Microsoft Visual Studio 8.
3. Set MSDEV_HOME=%VSINSTALLDIR%\Vc
4. Write *call "%VSINSTALLDIR%"\Common7\Tools\vsvars32.bat* at the prompt.
5. Ensure you have sufficient rights to execute .bat files.
6. Edit **compile_libcatia_tc_custom.bat** as required (enter the value for %TC_ROOT% corresponding to the Teamcenter installation location and %MSDEV_HOME%).
7. Execute the **compile_libcatia_tc_custom.bat** file. Administrator privileges may be required to perform this operation
8. Copy the **libcatia_tc_custom.lib** file to the *%TC_ROOT%\lib* directory.
9. Copy the **libcatia_tc_custom.dll** file to the *%TC_ROOT%\bin* directory.

7.1.4.3 Scripts

Refer to the following scripts for Windows or UNIX located in the *TcIC_DIR\sample* folder.

Scripts to compile/link with Teamcenter for **Windows**.

- compile_libcatia_tc_custom.bat
- compile.bat
- link_tc_custom_exits.bat

Scripts to call with Teamcenter for **UNIX**.

- compile_libcatia_tc_custom
- compile

- link_tc_custom_exits

7.2 RACLess Entry Point

RACLess Mode Through Server Support

Teamcenter Integration for CATIA V5 commands can be run without launching a connection through Teamcenter Rich Application Client (RAC). Running these commands without a RAC connection is called a **RACLess mode**.

The purpose of the RACLess mode is to be fully independent from the RAC application so that external applications can connect to TcIC commands using their own client application, such as Teamcenter Active Workspace or a customized client application, instead of using the Teamcenter Portal as the default client application.

TcIC can be run using a Teamcenter RAC client or in RACLess mode.

7.2.1 Definition of the client_env File

The client application which connects to the Integration is defined in the *%TCICV5_DIR%\env\client_env* file. By default, this file is set during installation of the Integration.

Teamcenter client communication system (TCCS) manages communication and file transfers between Teamcenter clients and servers. TCCS contains the FMS client cache (FCC), which uploads files from a workstation to a Teamcenter volume and also downloads requested files from the volume to a workstation.

The Integration supports a single host stand-alone TCCS installation defined in the *client_env* file (TCCS_SA option). TCCS and FCC file transfer is also supported. There is no Client installed.

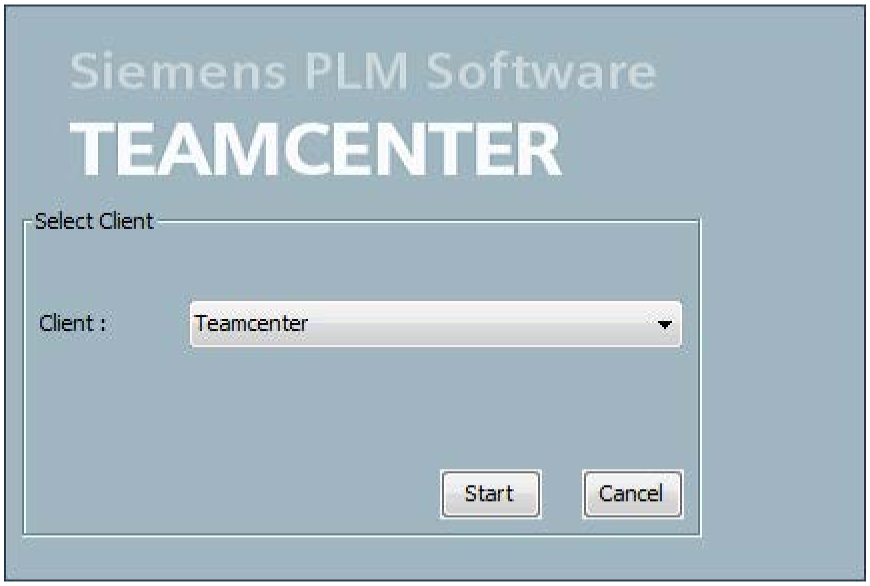
When the process launched is RAC/RACLess, the first client_0 defined in the *client_env* is used for the RACLess connection; when **-client** is not set or is invalid, the first valid client defined in the *client_env* file is used for the RACLess connection.

Clients in order of priority when connected to the RACLess Application are:

1. RAC
2. Teamcenter Loader (LOADER_CLIENT0)
3. Embedded Active Workspace (AW_CLIENT0)
4. Standalone Active Workspace (AW_CLIENT1)
5. Full RACLess (CATIA)

When a RACLess command launches and the client is not connected to the RACLess application, the first client available within the list of available client sessions, defined in *client_env* file, is used to connect to the RACLess application.

When the **Start Teamcenter** command launches and the client is not connected to the RACLess Application, a Teamcenter Client panel displays to select the appropriate client, from a list of available client sessions, to connect to the RACLess application.



The RACLess application launches:

- the socket server, defined by the TcIC_socket_port variable, to manage the interface between CATIA and the client (Portal, customized Loader, Active Workspace client…).
- a SOA connection to the Teamcenter server

The *%TCICV5_DIR%\env\client_env* file is structured as shown. Refer to the table below or click on the definitions for descriptions:

```
<client>
Client Name
Client Full Path
Category
Teamcenter Rich Client or FMS Directory
</client>
```

Definition	Description
Client Name	The client current configuration. It's strongly recommended to avoid having the same client name used twice in this file.
Client Full Path	The file used to launch the client (Teamcenter loader, portal.bat, fms…). For example, *portal_en.bat*
Category	Use **Client_0** when portal.bat (RAC) or TcICloader.bat (Teamcenter Loader) are the client. Interactive actions like Selection in Teamcenter and Refresh in the Structure Manager are supported, corresponding to a Standard Teamcenter Integration for CATIA V5 installation. Use **Client_1** when Active Workspace or Custom Loader are the client. In this case, interactive actions are not supported in Teamcenter Integration for CATIA V5. Use **TCCS_SA** for a stand-alone TCCS installation. Only full RACLess commands are supported because a client is not installed.
Teamcenter Rich Client or FMS Directory	Use **TC_PORTAL_ROOT** when the client is portal.bat with **Client_0** Use **TC_PORTAL_ROOT** or **FMS_HOME** for other clients (Active Workspace, Loader) . Use **FMS_HOME** when a client is not installed (with **TCCS_SA**).

Sample of a *client_env* file for Standard Teamcenter Integration for CATIA V5 (portal.bat client):

```
<client>
Teamcenter
C:\Tc10.1.2_20140617\portal\portal.bat
```

```
Client_0
C:\Tc10.1.2_20140617\portal
</client>
```

Sample of a *client_env* file for TCCS_SA:

```
<client>
NO RAC
TCCS_SA
C:\Tc10.1.2\tccs
</client>
```

Sample of a *client_env* file for Teamcenter Integration for CATIA V5 Loader client:

```
<client>
Teamcenter Integration for CATIA V5 Loader
C:\TCIC_V5\racless\TcICLoader.bat
Client_0
C:\Tc10.1.5_20151030\portal
</client>
```

Sample of a *client_env* file for Teamcenter Integration for CATIA Loader and Teamcenter Rich Client:

```
<client>
Teamcenter
C:\Tc10.1.5_20151030\portal\portal.bat
Client_0
C:\Tc10.1.5_20151030\portal
</client>
<client>
Teamcenter Integration for CATIA V5 Loader
C:\TCIC_V5\racless\TcICLoader.bat
Client_0
C:\Tc10.1.5_20151030\portal
</client>
```

Sample of a *client_env* file for Embedded Active Workspace client:

```
<client>
Teamcenter Embedded Active Workspace Client
http://vm-tcserver33:7031/awc
Client_0
C:\Tc10.1.5_20151030_4T_t1415e_vm-tcserver33\rac
</client>
```

When the Teamcenter Embedded Active Workspace Client has been manually added in the *client_env* file, ensure that the Windows registry has been updated with the *%TCICV5_DIR%\aw\aw_keys.reg* file and the **TcIC_awhosted_menu** variable in the %TCICV5_DIR%/env/tcic_var_env.xml file is set to **true**.

Sample of a *client_env* file for Standalone Active Workspace client:

```
<client>
Teamcenter Standalone Active Workspace Client
http://vm-tcserver33:7031/awc
Client_1
C:\Tc10.1.5_tccs_sa\tccs
</client>
```

When the Teamcenter Standalone Active Workspace Client has been manually added in the *client_env* file, ensure that the Windows registry is updated with the *%TCICV5_DIR%\aw\aw_keys.reg* file.

7.2.2 List of Functions Available in RACLess Mode

Key:

Full RACLess: A Teamcenter client is not running and a RACLess connection is established.
RAC only: A Teamcenter RAC client is running and a connection to the RACLess application is established.
RAC/RACLess: A Teamcenter RAC client is running and a RACLess connection is established.

Function	RAC/RACless
About integration	Full RACLess, RAC only, RACLess optional
Add CATShape Shape Representations	Full RACLess
Browse Multi-Model Links from CATIA	Full RACLess
Browse Multi-Model Links from Teamcenter	RAC / RACLess

Function	RAC/RACless
Check In	Full RACLess, RAC / RACLess
Check Out	Full RACLess, RAC / RACLess
Convert into JT	RAC / RACLess
Create CATPart dataset for selection	RAC / RACLess
Create CATProduct dataset for selection	RAC / RACLess
Create ExportSpreadsheet	RAC / RACLess
Create ImportSpreadsheet	Full RACLess
Detailed States	Full RACLess
Display mass properties in the top assembly origin	RAC / RACLess
Documentation	No connection required, RAC only
Export	RAC / RACLess
Export as reference	RAC / RACLess

Function	RAC/RACless
Highlight in CATIA	RAC / RACLess
Highlight in Teamcenter	RAC / RACLess
Import	Full RACLess
Insert	RAC / RACLess
Load	RAC / RACLess
Load as reference	RAC / RACLess
Load for Compare	RAC / RACLess
Load for Consulting	RAC / RACLess
Load from Teamcenter	RAC / RACLess
Load Merge	RAC / RACLess
Load Merge Selected Level	Full RACLess, RAC / RACLess
Load Merge Target	Full RACLess

Function	RAC/RACless
Load, Update, Save	RAC / RACLess
Loader	RACLess
Purge Staging Directory	No connection required
Read Linked Documents	Full RACLess
Refresh	Full RACLess
Remove links	RAC / RACLess
Replace by revision	Full RACLess
Replace from Teamcenter	RAC / RACLess
Replace load as cgr	Full RACLess
Return to CATIA V5	RAC only
Revise	RAC only
Save	Full RACLess

Function	RAC/RACless
Save As	Full RACLess, RAC only
Save Selected	Full RACLess
Spreadsheet Editor	Full RACLess
Start Teamcenter from CATIA	RAC only
Synchronize Active Shapes	Full RACLess
Update assembly mass properties	RAC / RACLess
Update status Icon in CATIA	Full RACLess
Update Title Block	Full RACLess
User settings from CATIA	Full RACLess
User Settings from Teamcenter	RACLess only
Volume search	RAC / RACLess

7.2.3 StartRACLessServer.bat file and Project Management

Launch the StartRACLessServer.bat file

Stop the RACLess Connection

The Teamcenter **projects** which are defined to provide a mechanism for organizing data and implementing access control based on project membership, can also be set during the RACLess connection and used to configure log in.

The client name defined in the *client_env* file is also set during the RACLess connection and used to configure log in. A cache file is created after the connection is made, stores the log in information and contains User/Group/Role/Host/TCCS parameters.

If the login operation fails, modify **TCIC_RETRY_LOGIN_COUNT** to define the number of connection attempts and modify **TCIC_RETRY_LOGIN_DELAY** to define the elapsed time (in seconds) between two connection attempts. By default, both values are 2.

The StartRACLessServer.bat file parameters are defined within the *%TCICV5_DIR%\racless* directory and support Teamcenter Projects (**-proj**) and Teamcenter Client name (**-client**).

When the **-proj** parameter is not set or is invalid, the default project is used for the RACLess connection, if it exists for the current user.

Set the **-proj** parameter without a value to permit the user to define the project for the connection. When **-proj** has been set without a value, no project is used for the RACLess connection.

The **-client** parameter defines the client name in the *client_env* file that is used when launching the command. When the process launched is RAC/RACLess, the first client_0 defined in the *client_env* is used for the RACLess connection; when **-client** is not set or is invalid, the first valid client defined in the *client_env* file is used for the RACLess connection.

Clients in order of priority when connected to the RACLess Application are:

1. RAC
2. Teamcenter Loader (LOADER_CLIENT0)
3. Embedded Active Workspace (AW_CLIENT0)

4. Standalone Active Workspace (AW_CLIENT1)

5. Full RACLess (CATIA)

7.2.3.1 Launch the StartRACLessServer.bat file

Use the following parameters to launch the StartRaclessServer.bat script.

```
-u=<user_id>
-p=<password>
-g=<group>
-r=<role>
-nl=<lang>
-h=<host> (for 2T:iiop:localhost:1572/TcServer1 or 4T: http://vm-
tcserver16:7025/tc or 4T TCCS: tccs_4T_t1312b_vm-tcserver16_sso)
-proj=<project_id>
-client=<client_name>
```

DOS versions interpret certain characters (for example, % < | >) before executing a command. Use double-quotes (") to avoid bad instructions.

When FMS_HOME is defined during the Installation process, JRE_HOME or JAVA_HOME variables are set in the StartRACLessServer.bat file, in the case those variables are defined.

7.2.3.2 Stop the RACLess connection

In some circumstances, the RACLess connection must be manually stopped to unlock CATIA/Teamcenter during troubleshooting.

There are two methods which can be used to deblock Teamcenter Integration processes to stop the RACLess connection.

- Create a bat file
- Teamcenter Integration System Tray

7.2.3.2.1 Create a .bat file

Create a .bat file containing the following instructions to stop the RACLess connection.

```
@echo off
REM Path of the jdk to use : to be updated if needed
```

```
set JDK_HOME=C:\Program Files\java\jdk1.7.0_17
C:
cd %JDK_HOME%\bin
jps -ml > test.racless.log
For /F "tokens=1,2" %%i in (test.racless.log) Do  IF
"%%j"=="com.ebsolutions.soacore.Launcher" taskkill /F /T /PID %%i
del test.racless.log
```

The following conditions should be confirmed before creating or running the .bat file.

1. A Java Development Kit (JDK) is installed on the machine.
2. Users have write access to the JDK directory. If they do not have write access, replace

   ```
   jps -ml > test.racless.log
   For /F "tokens=1,2" %%i in (test.racless.log) Do  IF
   ```

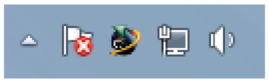
   with

   ```
   jps -ml > %TEMP%\test.racless.log
   For /F "tokens=1,2" %%i in (%TEMP%\test.racless.log) Do  IF "%%
   ```

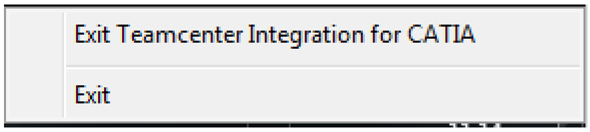
7.2.3.2.2 Teamcenter Integration System Tray

The Teamcenter Integration System Tray launches when a RACLess application launches. Use the system tray to control Teamcenter Integration processes. The System Tray is automatically closed when the RACLess application is closed.

The system tray is located at the bottom of the desktop, lower right-hand corner of the screen.



Use **Exit Teamcenter Integration for CATIA** to unlock Teamcenter Integration processes.



Exit Teamcenter Integration for CATIA

Exit

Remove the following lines in the *StarRACLessServer.bat* file to **deactivate** the system tray.

```
REM Start TcIC System Tray.
REM TcIC System Tray is optional. Comment following three lines to
turn it off.
set MAINCLASS=com.ebsolutions.systray.SystemTrayIconPopup
set SYSTRAY_CMD=START "TcIC System Tray" /B "%JRE_HOME%\bin\java" -
Xms256M -classpath "%RACLESS_CLASSPATH%" %MAINCLASS%
%SYSTRAY_CMD%
```

Manually launch the system tray by creating a .bat file containing following instructions.

```
@echo off
setlocal
REM
*******************************************************************
*************
REM This script launches a SYSTRAY application.
REM Set JRE_HOME
REM If value JRE_HOME contains spaces, you must enclose the string
in double quotes
REM Please ensures %JAVA_HOME% defined in CATIA environment file
match the expected Java version.
REM Otherwise it may create conflicts and prevent CATIA integration
to work.
REM if defined JAVA_HOME set JRE_HOME=%JAVA_HOME%
if defined JRE_HOME set JRE_HOME=%JRE_HOME%
if defined JRE64_HOME set JRE_HOME=%JRE64_HOME%
if not defined JRE_HOME set JRE_HOME=C:\Program Files\Java\jre7
REM set the classpath
REM Specify a list of directories, JAR archives, and ZIP archives
to search for.
REM Class path entries are separated by semicolons (;). Specifying
SYSTRAY_CLASSPATH overrides any setting of the CLASSPATH
environment variable.
REM If the path entry contains spaces; you must enclose the string
in double quotes.
REM define RACLESS_DIR
set RACLESS_DIR=%~dp0
set
SYSTRAY_CLASSPATH=%RACLESS_DIR%Systray.jar;%RACLESS_DIR%lib\swt.jar
set PATH=%PATH%;%FMS_HOME%\bin;%FMS_HOME%\lib;%ORIGINAL_PATH%
REM java -classpath "%SYSTRAY_CLASSPATH%" -cp
C:\TCIC_V5\racless\Systray.jar
com.ebsolutions.systray.SystemTrayIconPopup
REM java -classpath "%SYSTRAY_CLASSPATH%" -jar Systray.jar
set MAINCLASS=com.ebsolutions.systray.SystemTrayIconPopup
set SYSTRAY_CMD= "%JRE_HOME%\bin\java" -Xms256M -classpath
"%SYSTRAY_CLASSPATH%" %MAINCLASS% %*
%SYSTRAY_CMD%
```

Then use the **Exit** command to manually exit the System Tray.

Ensure that the following path is defined in your PATH environment variable:
`%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem`

7.2.4 JAVA Code Customizations

This section contains methods, pre/post actions and steps required to customize the RACLess processes used within the Teamcenter Integration for CATIA V5 using JAVA SOA RACLess APIs.

7.2.4.1 Introduction and Implementations

7.2.4.1.1 Prerequisites, Folder Hierarchy and File Descriptions

[Prerequisites](#)

[Folder Hierarchy and File Definitions](#)

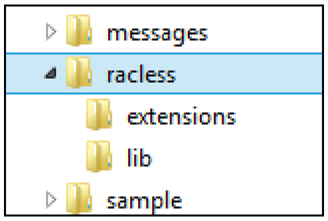
Samples described in this guide were built using Eclipse version 3.8.0. RACLess APIs have been developed in JAVA SOA technologies.

Prerequisites

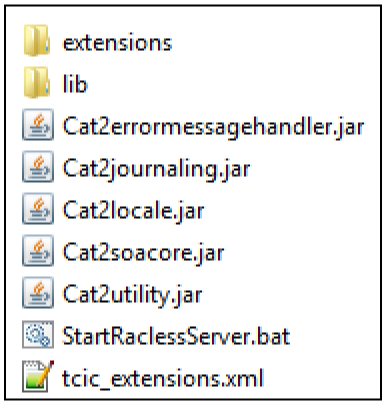
- Strong knowledge of JAVA SOA technologies is required.
- JAVA version 1.6.0_24 is required with Teamcenter 9.1.2.9.
- JAVA version 1.7.0_17 is required with Teamcenter 10.1.2.

Folder Hierarchy and File Descriptions

RACLess APIs are available within the *%TCICV5_DIR%\racless* directory, sub-directories and libraries.



```
messages
racless
    extensions
    lib
sample
```



```
extensions
lib
Cat2errormessagehandler.jar
Cat2journaling.jar
Cat2locale.jar
Cat2soacore.jar
Cat2utility.jar
StartRaclessServer.bat
tcic_extensions.xml
```

Files

The *Cat2errormessagehandler.jar, Cat2journaling.jar, Cat2locale.jar,* and *Cat2utility.jar* files are libraries for internal use only.

The *Cat2soacore.jar* file contains the ICatiaUserExit and ICommandEvent APIs definitions and libraries to enable:

- customizing an existing process.

- creating a user process.
- Interface implementations.

The *StartRaclessServer.bat* file launches the RACLess application.

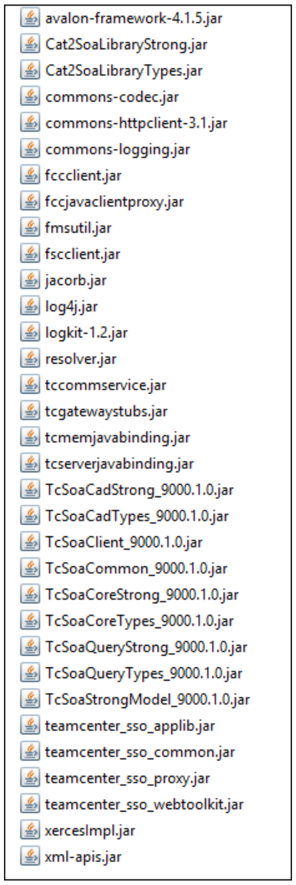
The *tcic_extensions.xml* file contains the declaration of the ICatiaUserExit and ICommandEvent APIs implementations.

Folders

The *extensions* sub-directory contains user libraries:

- Pre/post actions definitions
- ICatiaUserExit RACLess
- ICommandEvent user implementation

The *lib* sub-directory contains the files for **Teamcenter 9.1.2.9** SOA Strong.



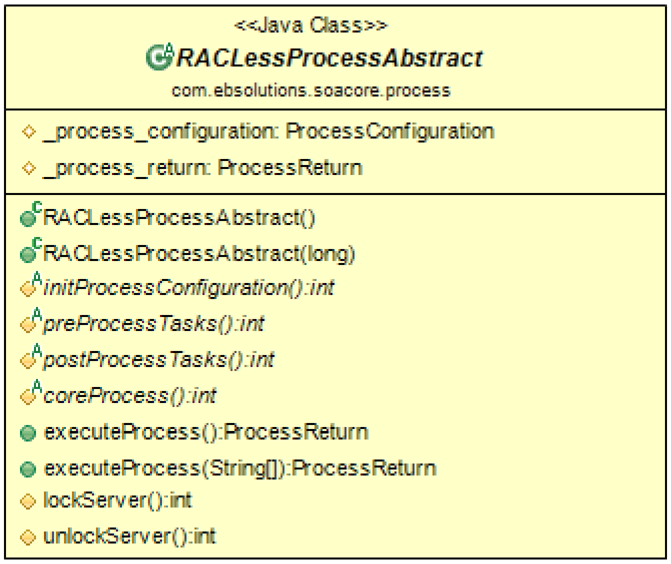
```
avalon-framework-4.1.5.jar
Cat2SoaLibraryStrong.jar
Cat2SoaLibraryTypes.jar
commons-codec.jar
commons-httpclient-3.1.jar
commons-logging.jar
fccclient.jar
fccjavaclientproxy.jar
fmsutil.jar
fscclient.jar
jacorb.jar
log4j.jar
logkit-1.2.jar
resolver.jar
tccommservice.jar
tcgatewaystubs.jar
tcmemjavabinding.jar
tcserverjavabinding.jar
TcSoaCadStrong_9000.1.0.jar
TcSoaCadTypes_9000.1.0.jar
TcSoaClient_9000.1.0.jar
TcSoaCommon_9000.1.0.jar
TcSoaCoreStrong_9000.1.0.jar
TcSoaCoreTypes_9000.1.0.jar
TcSoaQueryStrong_9000.1.0.jar
TcSoaQueryTypes_9000.1.0.jar
TcSoaStrongModel_9000.1.0.jar
teamcenter_sso_applib.jar
teamcenter_sso_common.jar
teamcenter_sso_proxy.jar
teamcenter_sso_webtoolkit.jar
xercesImpl.jar
xml-apis.jar
```

7.2.4.1.2 Customized Implementations

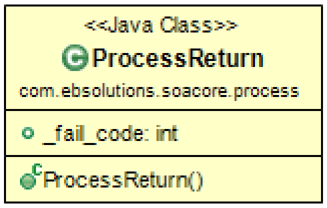
Introduction

New processes can be implemented and called during the RACLess application. Each RACLess application class inherits from the **RACLessProcessAbstract** abstract class.

The following methods and variables are available for this class. Other functions and variables not listed are dedicated for internal use only.


<<Java Class>>
RACLessProcessAbstract
com.ebsolutions.soacore.process
_process_configuration: ProcessConfiguration
_process_return: ProcessReturn
RACLessProcessAbstract()
RACLessProcessAbstract(long)
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int
executeProcess():ProcessReturn
executeProcess(String[]):ProcessReturn
lockServer():int
unlockServer():int

The **executeProcess** function is the main function called by the RACLess application. The executeProcess function manages events defined in iCommandEvent, ICatiaUserExit entry points, pre/post actions and initialization required for RACLess processes. The executeProcess function should always return an instance of ProcessReturn, for example, the _process_return variable member of the RACLessProcessAbstract class.



There are two ways to customize:

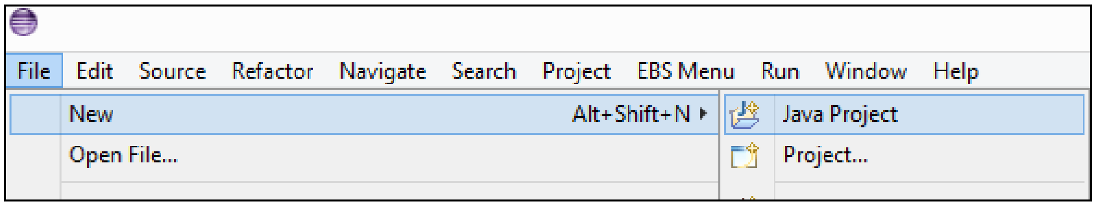
1. Overload the executeProcess or executeProcess(String[]) functions of all abstracted classes so as not to be dependent on the ICommandEvent class or other RACLess processes.

2. Overload the fourth abstract function to reuse the ICommandEvent management or pre/post actions entry points.

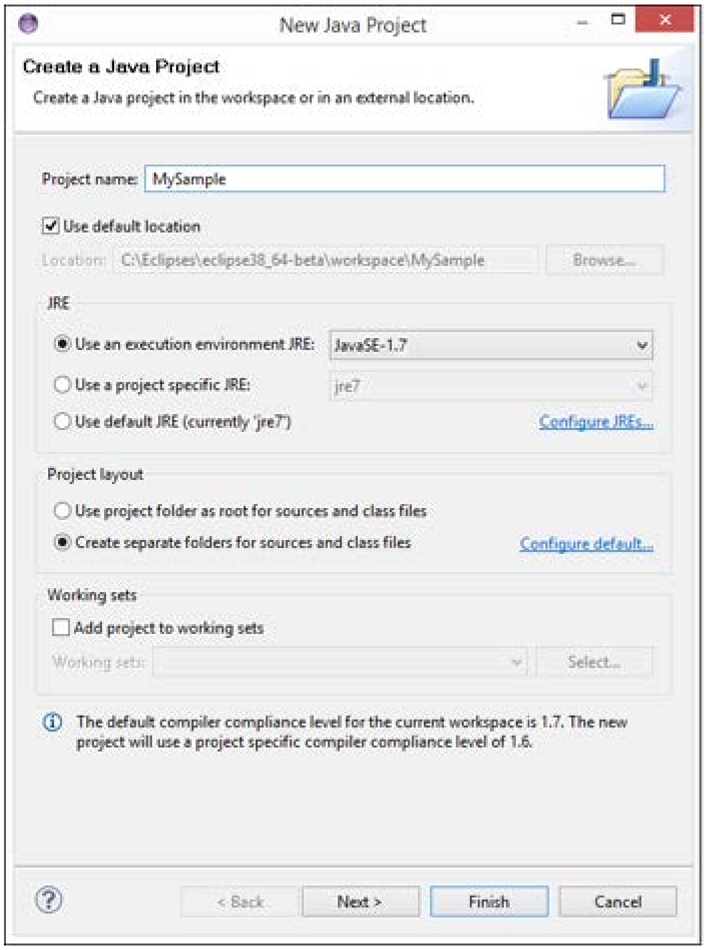
A customized process must always be placed into a specific package such as com.MyCusto

The Overload executeProcess Method (Method 1)

1. In Eclipse, create a new Java project.



2. Enter the **Project name** then select the compatible JRE version: **Java 1.7** for Teamcenter 10.1.2.5 and Teamcenter 11.2.1. Pick **Next** to continue.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MySample

[✓] Use default location

Location: C:\Eclipses\eclipse38_64-beta\workspace\MySample

JRE

(●) Use an execution environment JRE: JavaSE-1.7

( ) Use a project specific JRE: jre7

( ) Use default JRE (currently 'jre7')

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

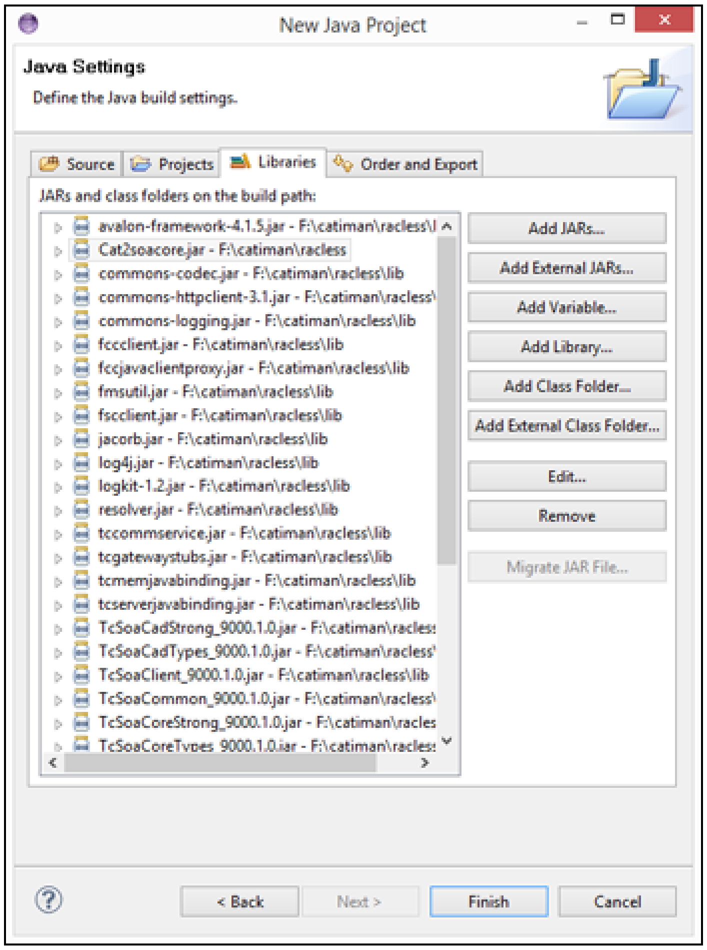
Working sets:

The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.6.

3. Select the **Libraries** tab, then select the libraries below required for the project. Pick **Finish** when done.

   a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

   b. Cat2soacore.jar library contained in *%TCICV5_DIR%\racless.*



New Java Project

Java Settings

Define the Java build settings.

Source | Projects | Libraries | Order and Export

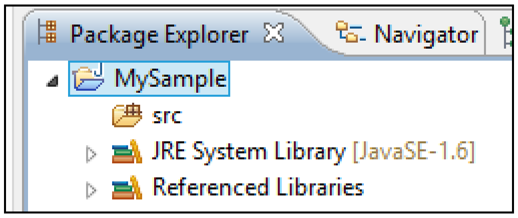
JARs and class folders on the build path:

```
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless
```

Add JARs... | Add External JARs... | Add Variable... | Add Library... | Add Class Folder... | Add External Class Folder... | Edit... | Remove | Migrate JAR File...

< Back | Next > | Finish | Cancel

Result:



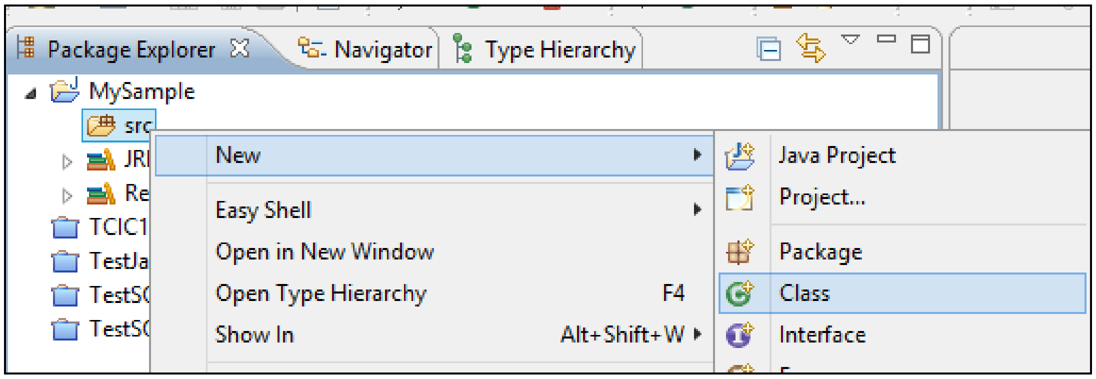
Package Explorer | Navigator

```
MySample
    src
  JRE System Library [JavaSE-1.6]
  Referenced Libraries
```

Overload the executeProcess function

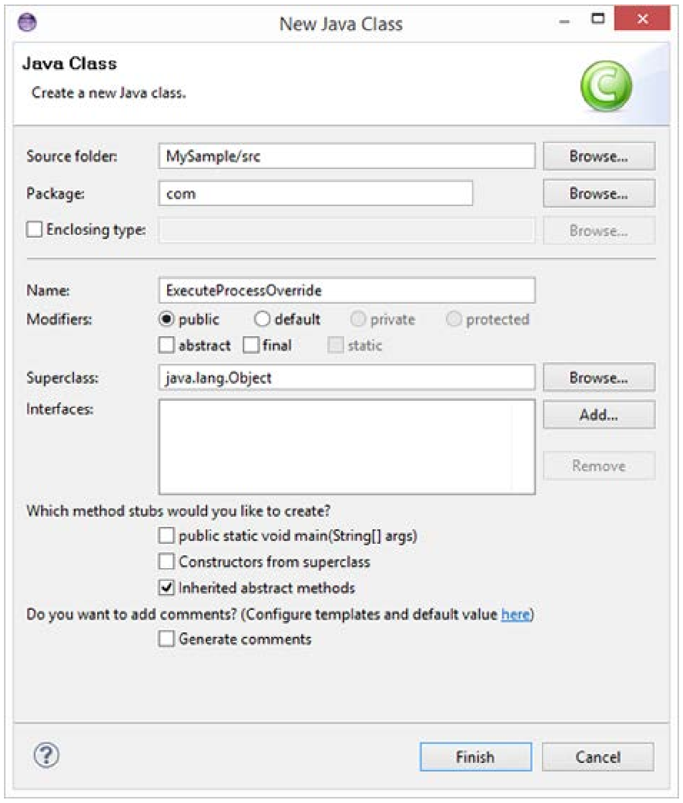
A customized process must always be placed into a specific package such as com.MyCusto

4. Select the **src** folder, right-click and then select **New >> Class**.



5. Complete the **Package:** and **Name:** fields on the New Java Class window.

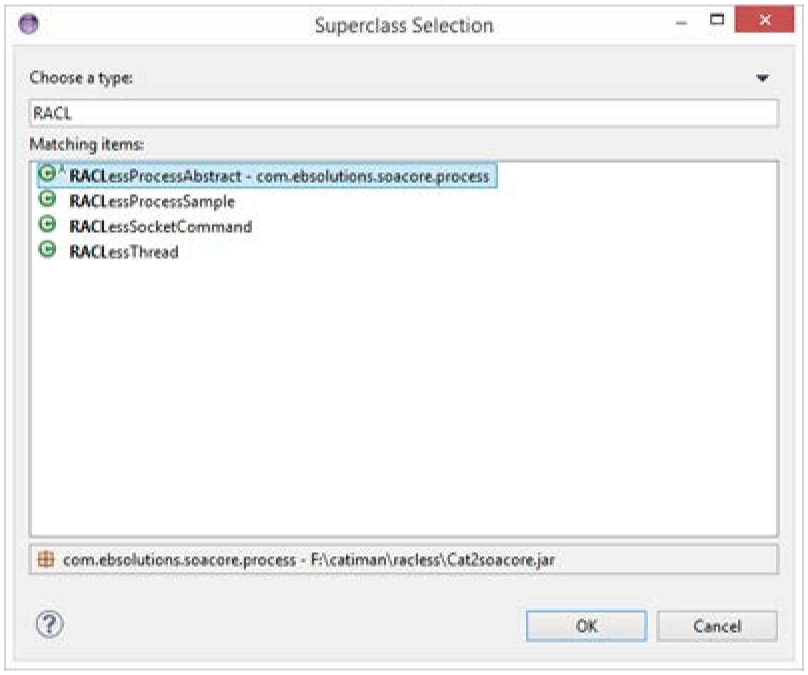
6. Untoggle **public static void main(String[] args)** and toggle the **Inherited abstract methods** option on. Pick the **Browse...** button next to the Superclass: field.



Java Class
Create a new Java class.

Source folder: MySample/src
Package: com
☐ Enclosing type:
Name: ExecuteProcessOverride
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

7. Enter *RACLessProcessAbstract* in the Choose a type: field. Select **RACLessProcessAbstract** from the resulting Matching items box, then pick **OK**.

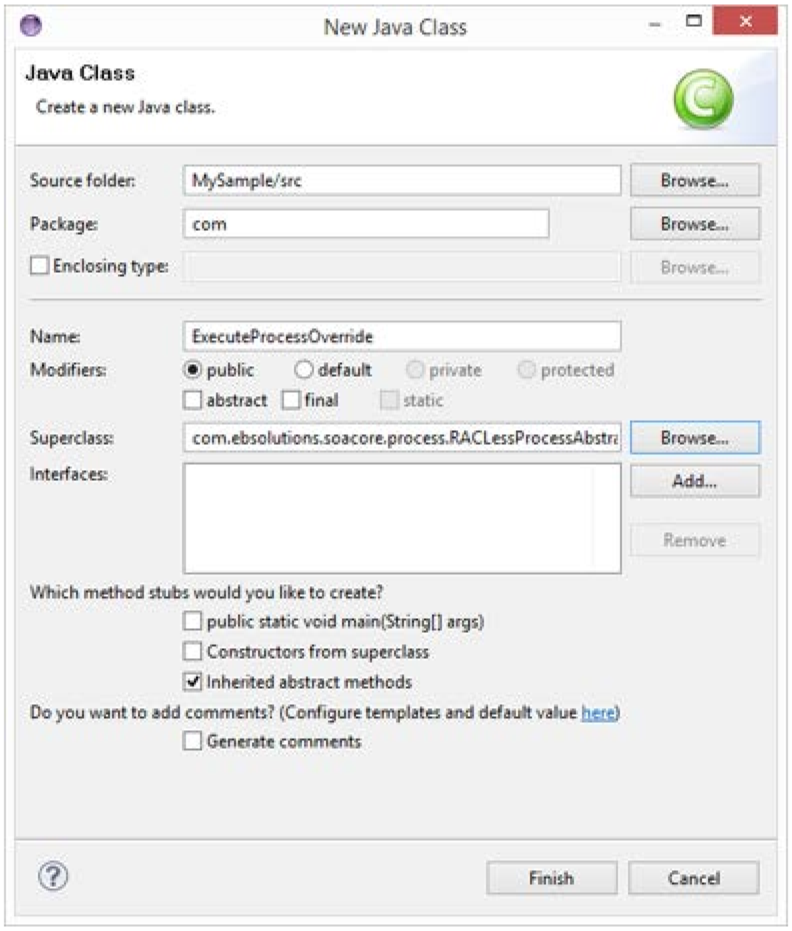


RACL

RACLessProcessAbstract - com.ebsolutions.soacore.process
RACLessProcessSample
RACLessSocketCommand
RACLessThread

com.ebsolutions.soacore.process - F:\catiman\racless\Cat2soacore.jar

8. Pick **Finish**.



Java Class

Create a new Java class.

Source folder: MySample/src

Package: com

☐ Enclosing type:

Name: ExecuteProcessOverride

Modifiers: ● public ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: com.ebsolutions.soacore.process.RACLessProcessAbstr

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

✓ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

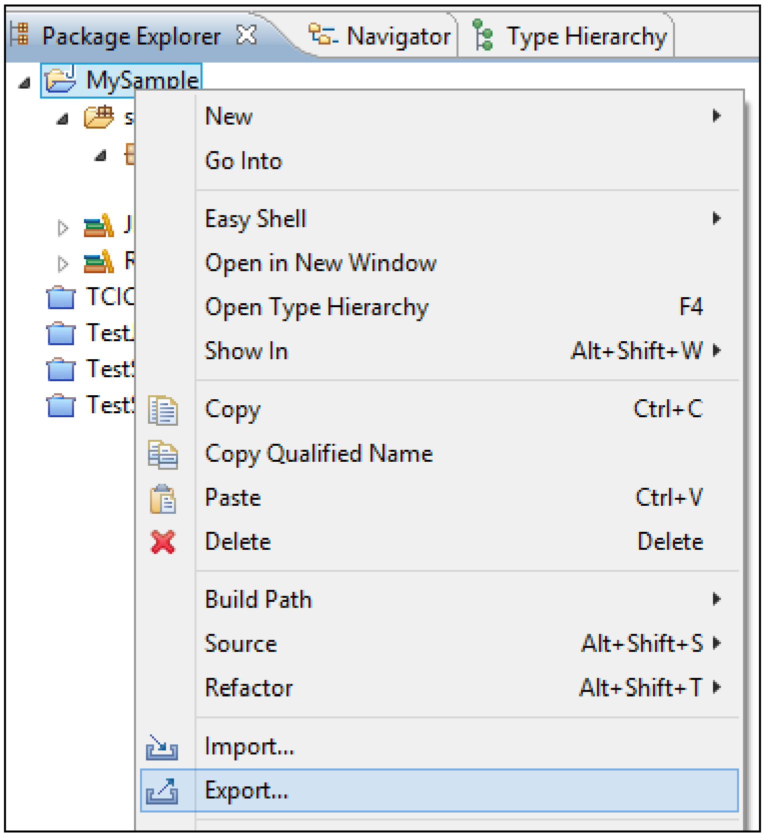
☐ Generate comments

9. Add a new executeProcess function without parameters. The _process_return is set to OK status, by default, indicating the process is successful.

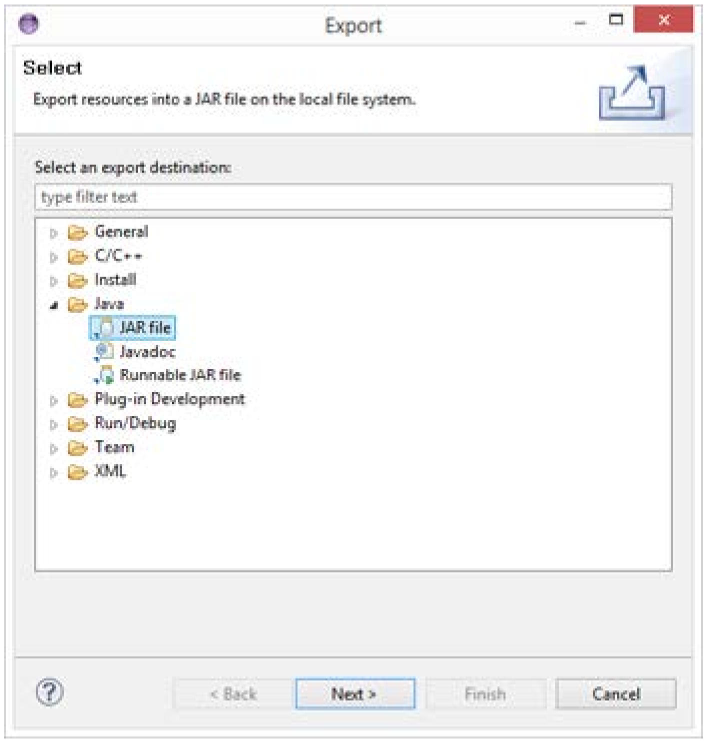
```
@Override
       public ProcessReturn executeProcess() throws Exception
       {
             JOptionPane.showMessageDialog(null, "my message");
             return _process_return;
       }
```

Generate the Library

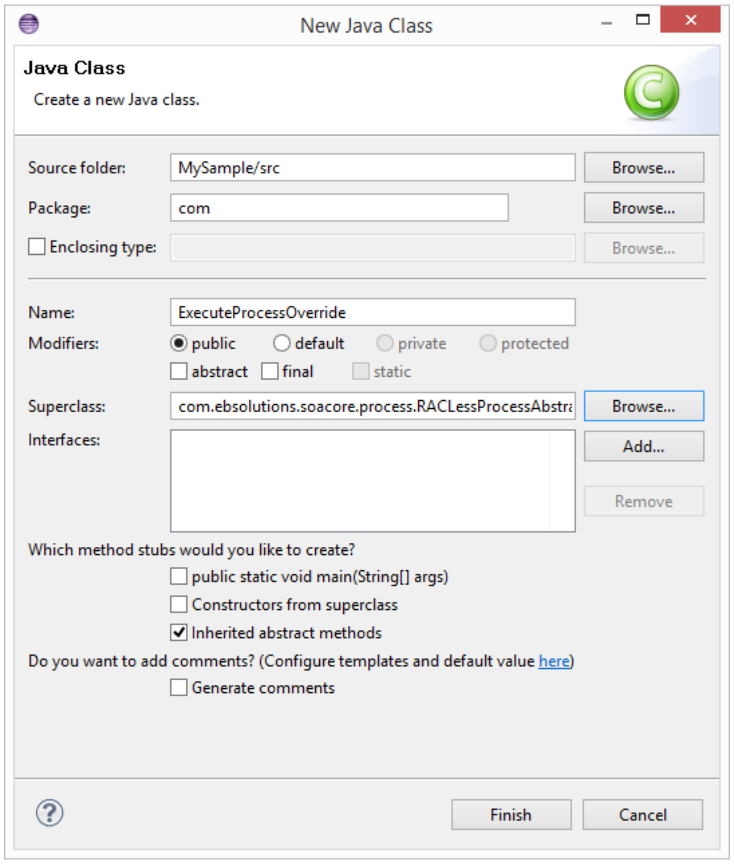
10. Highlight the project folder, then right-click and pick **Export**.



11. Toggle the **Java** folder open, select *JAR file*, then pick **Next**.



12. Select the project then enter a name and library location in the **JAR file** field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.

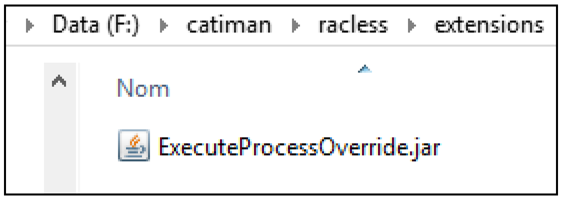

New Java Class

Java Class
Create a new Java class.

Source folder: MySample/src
Package: com
[ ] Enclosing type:
Name: ExecuteProcessOverride
Modifiers: (●) public ( ) default ( ) private ( ) protected
[ ] abstract [ ] final [ ] static
Superclass: com.ebsolutions.soacore.process.RACLessProcessAbstra
Interfaces:

Which method stubs would you like to create?
[ ] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

The library automatically loads when the RACLess process launches.


Data (F:) ▸ catiman ▸ racless ▸ extensions

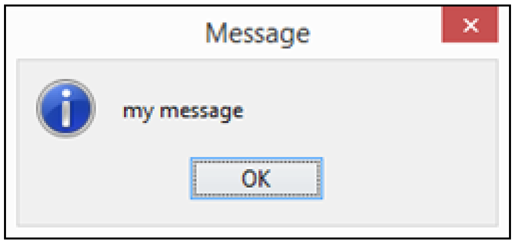
Nom

ExecuteProcessOverride.jar

Test the Implementation

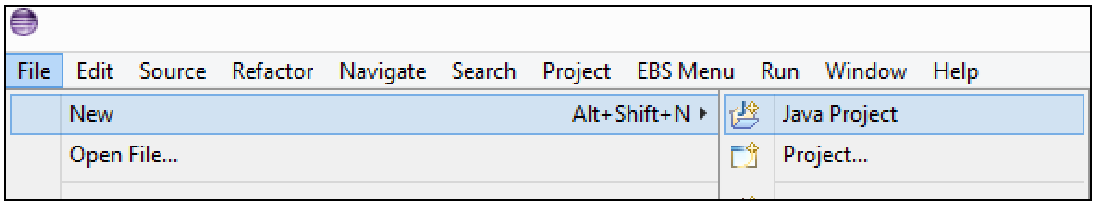
1. Use the *%TCICV5_DIR%\racless\StartRaclessServer.bat* command.

2. Launch following instruction in a command line: *%TCICV5_DIR%\bin\WIN32\SocketV5.exe com.ExecuteProcessOverride 0 11400.* The following message displays:

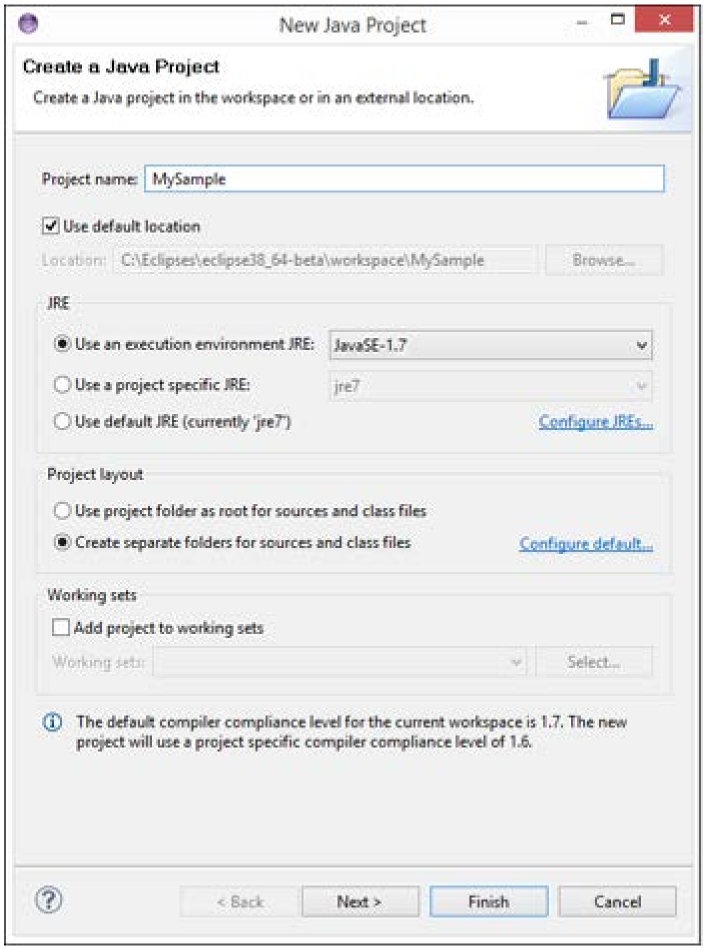


The Overload Abstract Class Method (Method 2)

1. In Eclipse, create a new Java project.



2. Set the **Project name** then select the compatible JRE version: **Java 1.7** for Teamcenter 10.1.2.5 and Teamcenter 11.2.1. Pick **Next** to continue.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MySample

[✓] Use default location

Location: C:\Eclipses\eclipse38_64-beta\workspace\MySample

JRE

[●] Use an execution environment JRE: JavaSE-1.7

[ ] Use a project specific JRE: jre7

[ ] Use default JRE (currently 'jre7')

Project layout

[ ] Use project folder as root for sources and class files

[●] Create separate folders for sources and class files

Working sets

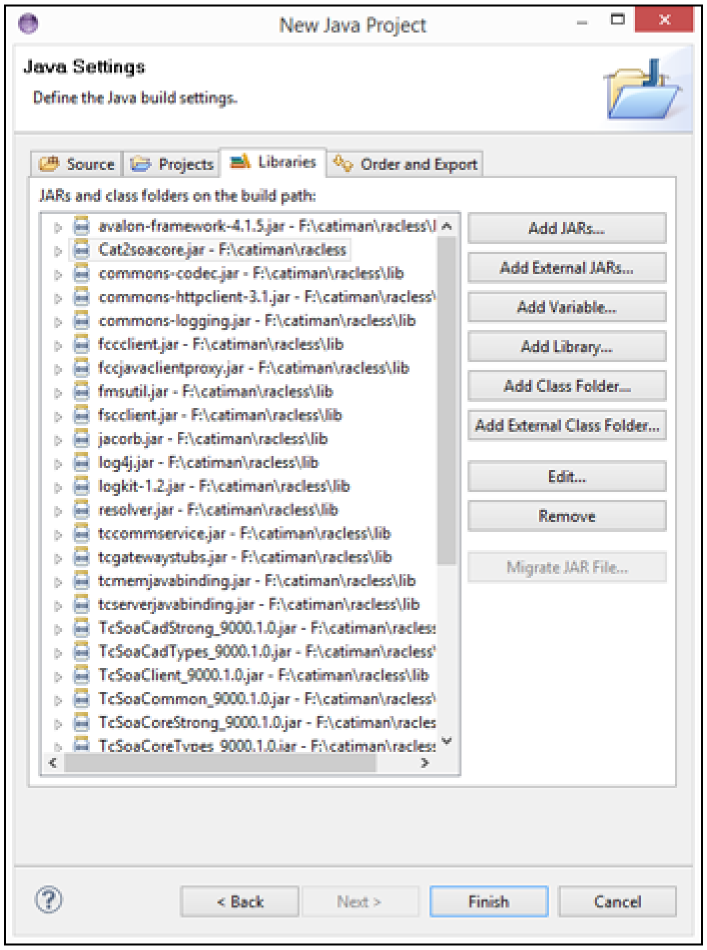
[ ] Add project to working sets

The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.6.

3. Select the **Libraries** tab, then select the following libraries required for the project. Pick **Finish** when done.

   a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

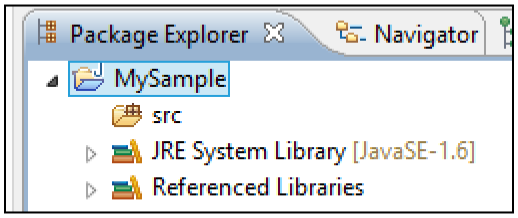
   b. Cat2soacore.jar library contained in *%TCICV5_DIR%\racless.*



JARs and class folders on the build path:

```
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless
```

Result:

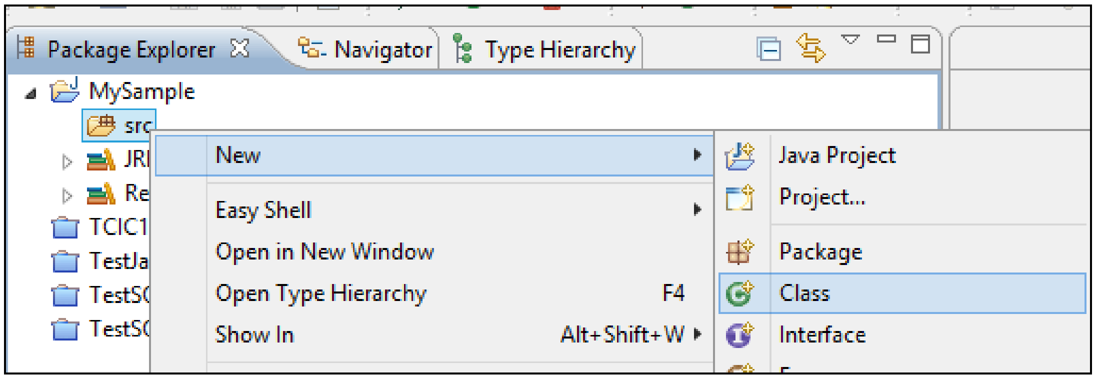


```
MySample
    src
    JRE System Library [JavaSE-1.6]
    Referenced Libraries
```

Overload the abstract class

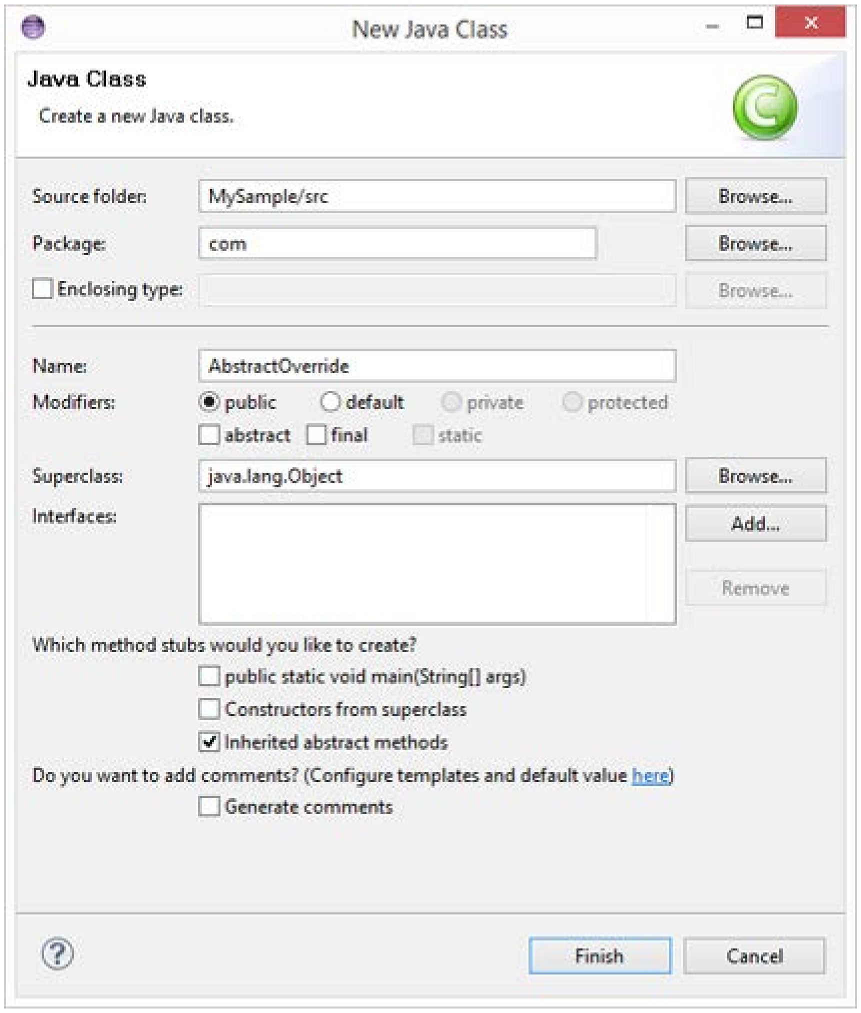
A customized process must always be placed into a specific package such as com.MyCusto

4. Select the **src** folder, right-click and then select **New >> Class**.



5. Complete the **Package:** and **Name:** fields.

6. Untoggle **public static void main(String[] args)** and toggle the **Inherited abstract methods** option on. Pick the **Browse...** button next to the Superclass: field.


New Java Class

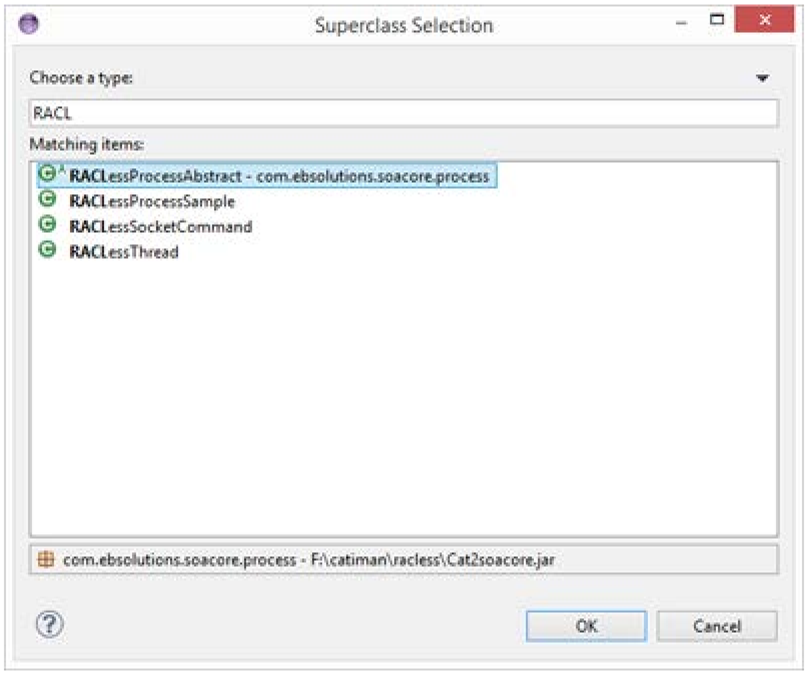
Java Class
Create a new Java class.

Source folder: MySample/src
Package: com
☐ Enclosing type:

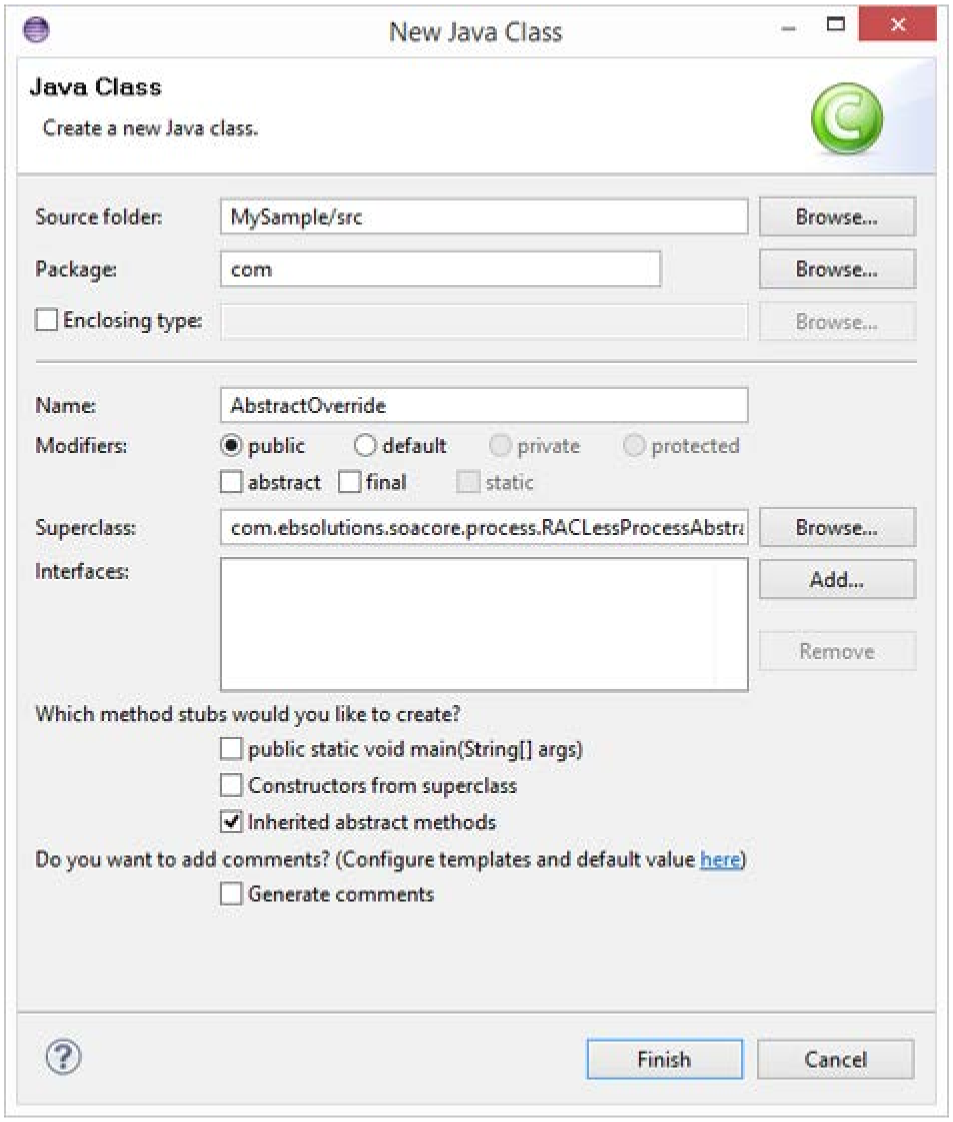
Name: AbstractOverride
Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

7. Enter *RACLessProcessAbstract* in the Choose a type field. Select **RACLessProcessAbstract** from the resulting Matching Items box, then pick **OK**.


Superclass Selection
Choose a type:
RACL
Matching items:
RACLessProcessAbstract - com.ebsolutions.soacore.process
RACLessProcessSample
RACLessSocketCommand
RACLessThread
com.ebsolutions.soacore.process - F:\catiman\racless\Cat2soacore.jar
OK
Cancel

8. Pick **Finish**.


New Java Class

Java Class

Create a new Java class.

Source folder: MySample/src

Package: com

☐ Enclosing type:

Name: AbstractOverride

Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static

Superclass: com.ebsolutions.soacore.process.RACLessProcessAbstra

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

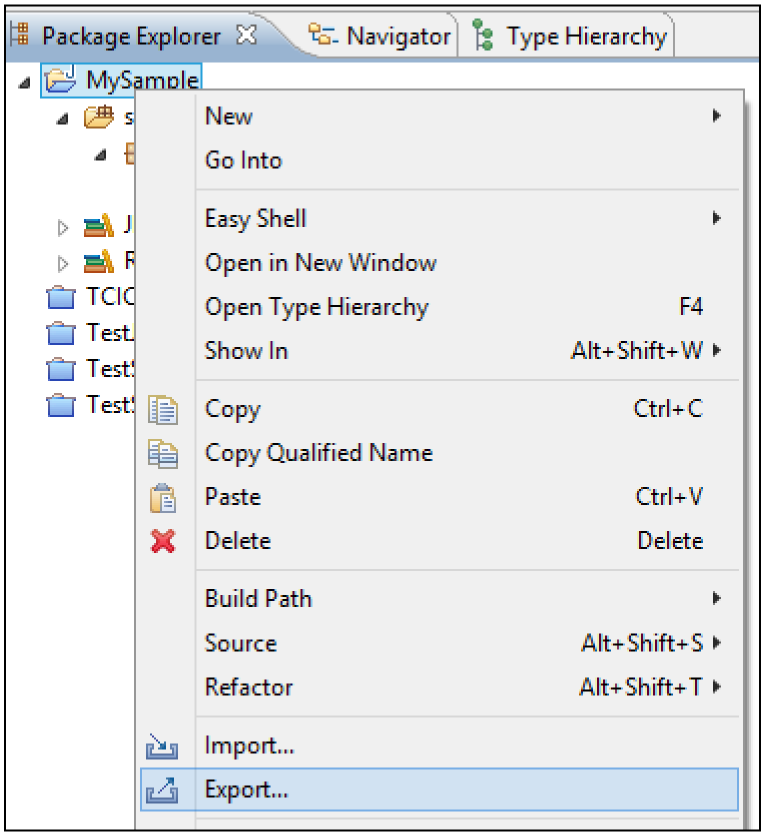
9. Implement the function as follows:

```
@Override
protected int coreProcess() {
   JOptionPane.showMessageDialog(null, "coreProcess message");
   return 0;
}

@Override
protected int initProcessConfiguration() {
      _process_configuration =
ProcessConfiguration.getDefaultProcessConfiguration();
```

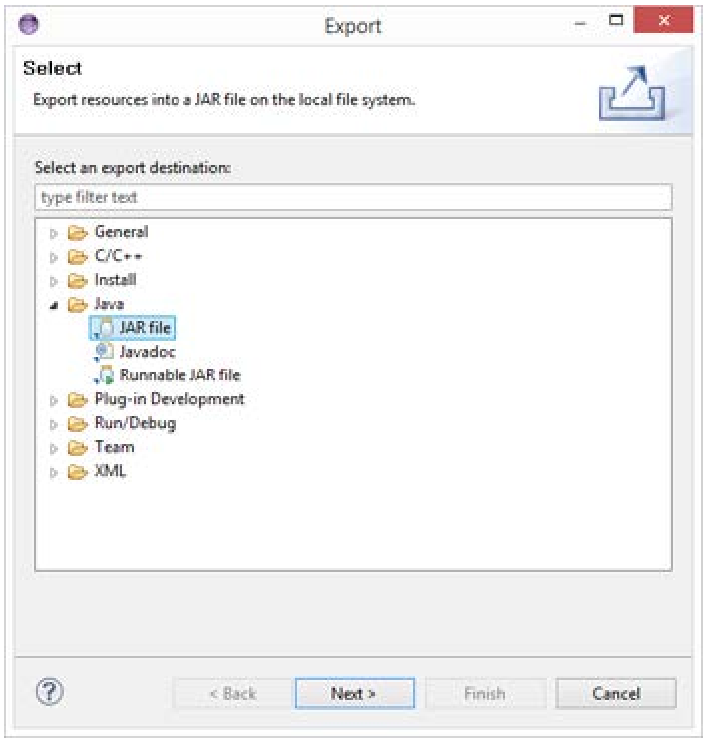
```
		return 0;
}

@Override
protected int postProcessTasks() {
		JOptionPane.showMessageDialog(null, "postProcess message");
		return 0;
}

@Override
protected int preProcessTasks() {
		JOptionPane.showMessageDialog(null, "preProcess message");
		return 0;
}
```

Generate the Library

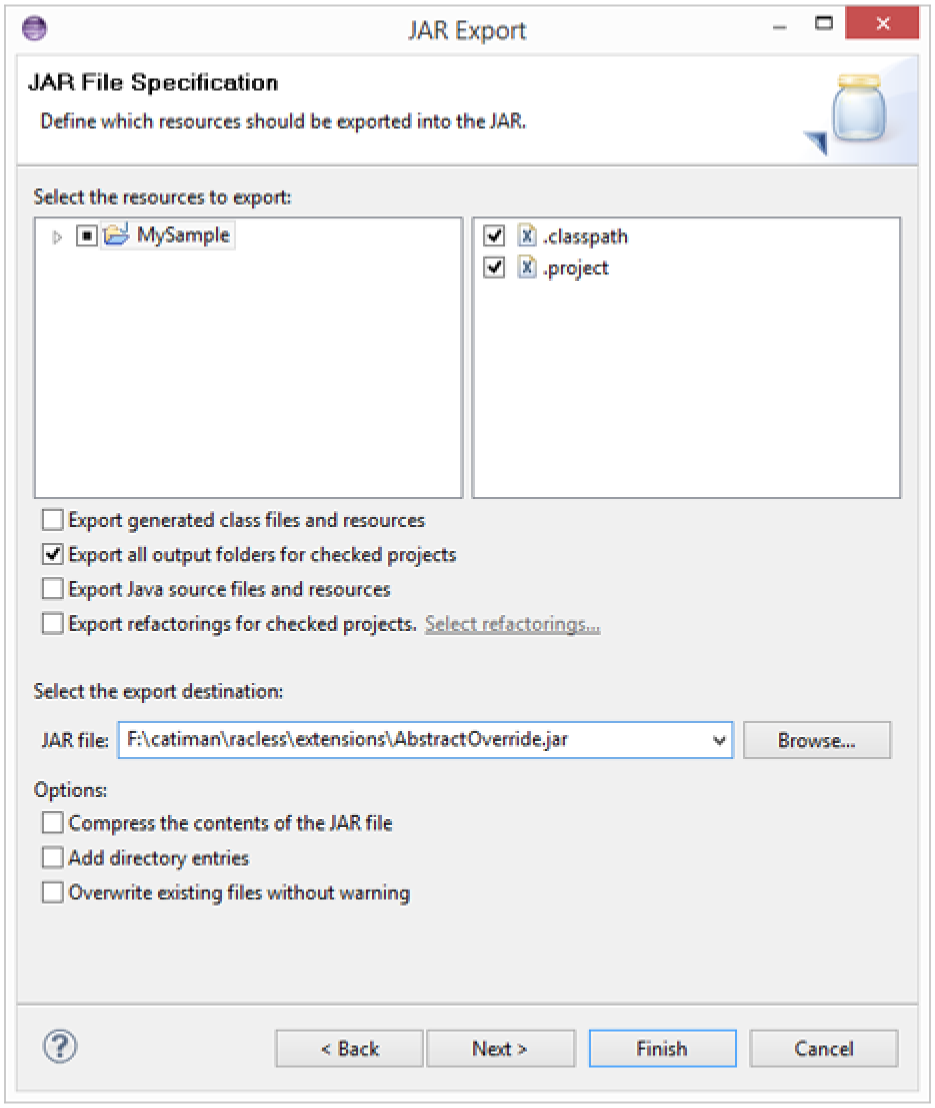
10. Highlight the project folder, then right-click and pick **Export**.



11. Toggle the **Java** folder open, select *JAR file*, then pick **Next**.



12. Select the project then enter a name and library location in the **JAR file** field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [■] MySample
- [✓] .classpath
- [✓] .project

- [ ] Export generated class files and resources
- [✓] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

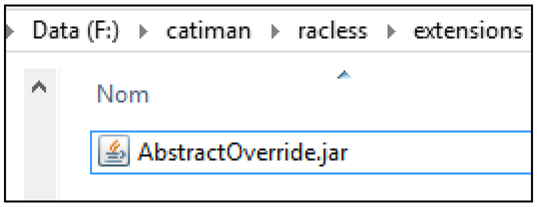
Select the export destination:

JAR file: F:\catiman\racless\extensions\AbstractOverride.jar

Options:

- [ ] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

The library automatically loads when the RACLess process launches.


Data (F:) ▸ catiman ▸ racless ▸ extensions

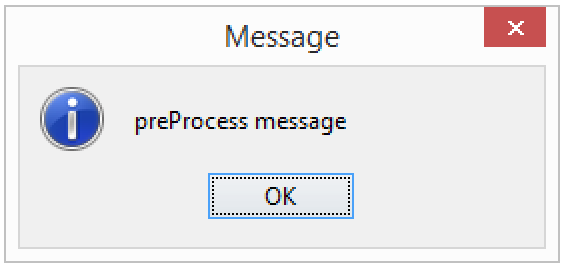
Nom

AbstractOverride.jar

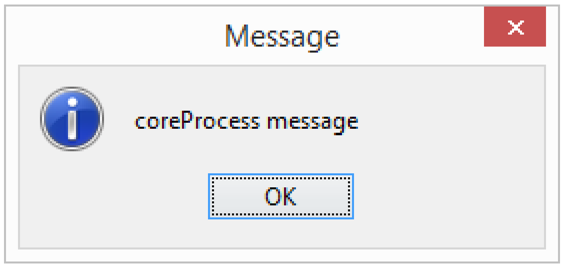
Test the Implementation

1. Use the *%TCICV5_DIR%\racless\StartRaclessServer.bat* command.

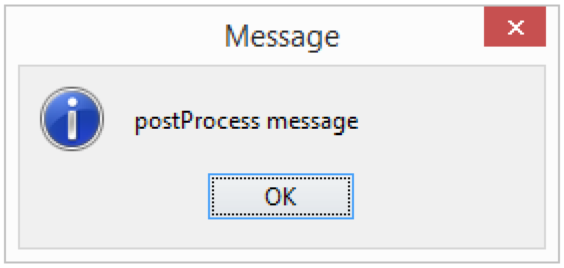
2. Launch following instruction in a command line *%TCICV5_DIR%\bin\WIN32\SocketV5.exe com.AbstractOverride 0 11400.* The following messages displays:



preProcess message



coreProcess message



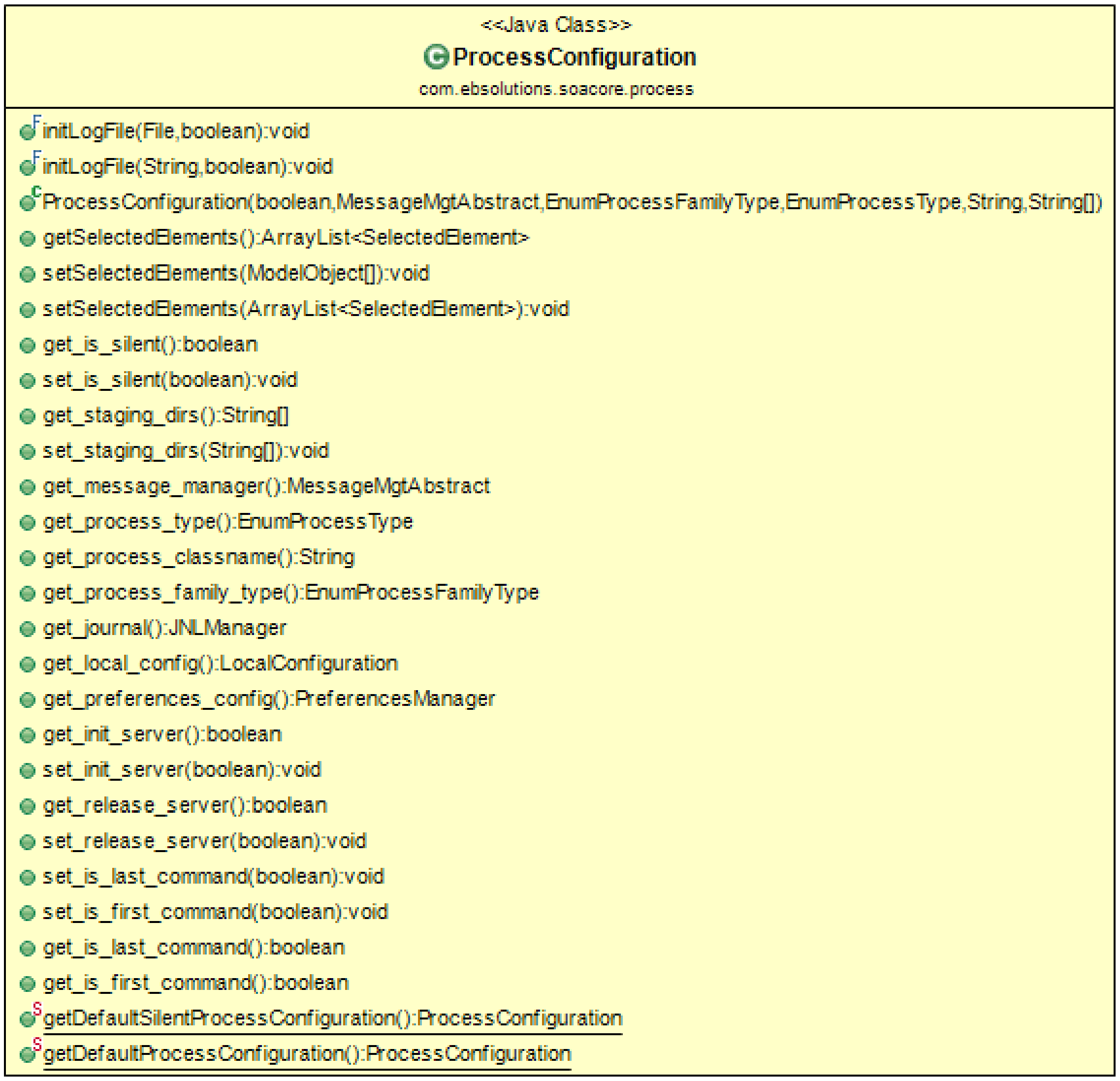
postProcess message

ProcessConfiguration Class

The configuration of each process must be defined through the ProcessConfiguration class. This configuration is set in the **_process_configuration** variable. Modify the configuration to change messages without having to modify the main code.

It's recommended to use the default configurations using the following methods:

- getDefaultSilentProcessConfiguration() for silent processes when the display message method is not used
- getDefaultProcessConfiguration() for silent processes when the display message method is used.



ProcessConfiguration

com.ebsolutions.soacore.process

- initLogFile(File,boolean):void
- initLogFile(String,boolean):void
- ProcessConfiguration(boolean,MessageMgtAbstract,EnumProcessFamilyType,EnumProcessType,String,String[])
- getSelectedElements():ArrayList<SelectedElement>
- setSelectedElements(ModelObject[]):void
- setSelectedElements(ArrayList<SelectedElement>):void
- get_is_silent():boolean
- set_is_silent(boolean):void
- get_staging_dirs():String[]
- set_staging_dirs(String[]):void
- get_message_manager():MessageMgtAbstract
- get_process_type():EnumProcessType
- get_process_classname():String
- get_process_family_type():EnumProcessFamilyType
- get_journal():JNLManager
- get_local_config():LocalConfiguration
- get_preferences_config():PreferencesManager
- get_init_server():boolean
- set_init_server(boolean):void
- get_release_server():boolean
- set_release_server(boolean):void
- set_is_last_command(boolean):void
- set_is_first_command(boolean):void
- get_is_last_command():boolean
- get_is_first_command():boolean
- getDefaultSilentProcessConfiguration():ProcessConfiguration
- getDefaultProcessConfiguration():ProcessConfiguration

Some of the accessors are for internal use only, but are available to view their values.

getDefaultSilentProcessConfiguration()

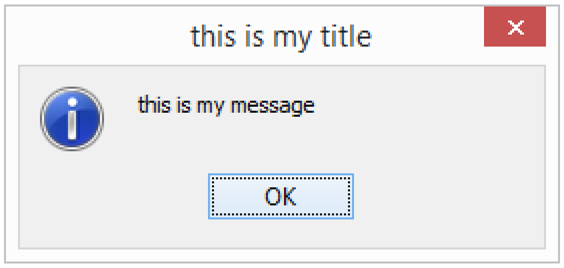
The following implementation does not display a message and traces are written to the *DefaultSilentCommand.log* file.

```
@Override
protected int coreProcess() {
    MessageMgtAbstract msg_manager = null;
    msg_manager = _process_configuration.get_message_manager();
    msg_manager.showMessage("this is my message",
                 "this is my title", 1);
    return 0;
}

@Override
protected int initProcessConfiguration() {
     _process_configuration =
     ProcessConfiguration.getDefaultSilentProcessConfiguration();
     return 0;
}
```

getDefaultProcessConfiguration()

The sample code shown above displays the following message:

```
@Override
protected int initProcessConfiguration() {
      _process_configuration =
ProcessConfiguration.getDefaultProcessConfiguration();
      return 0;
}
```



this is my message

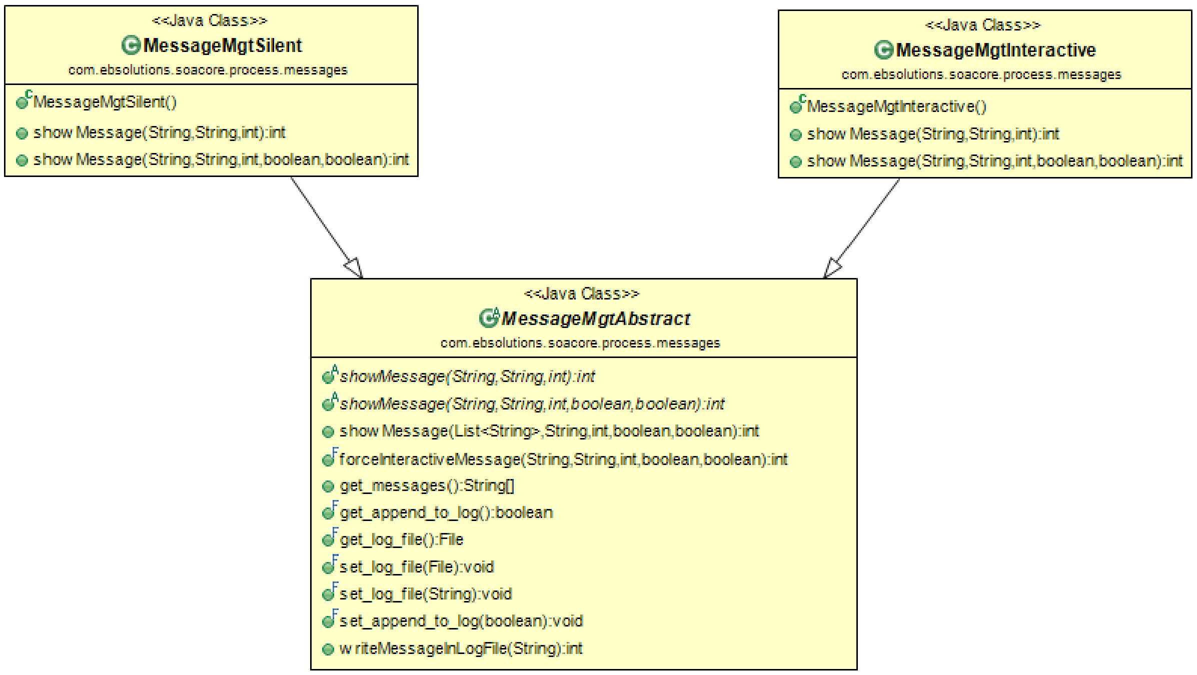
Customize the Log File

Use the **InitLogFile** function to define a new log file. In this sample, the log file is created under the *%WORK_DIR%\tmp* directory and named *my_process.log*.

```
@Override
protected int initProcessConfiguration()
{
     _process_configuration =
ProcessConfiguration.getDefaultSilentProcessConfiguration();
     _process_configuration.initLogFile("my_process.log", false);
     return 0;
}
```

MessageMgtAbstract

Each process manages a configuration like ProcessConfiguration. A message manager exists within each configuration and enables messages to be displayed in a dialog/message box or in the log file through the **MessageMgtAbstract**, **MessageMgtInteractive** and **MessageMgtSilent** classes.


<<Java Class>>
MessageMgtSilent
com.ebsolutions.soacore.process.messages
MessageMgtSilent()
showMessage(String,String,int):int
showMessage(String,String,int,boolean,boolean):int
<<Java Class>>
MessageMgtInteractive
com.ebsolutions.soacore.process.messages
MessageMgtInteractive()
showMessage(String,String,int):int
showMessage(String,String,int,boolean,boolean):int
<<Java Class>>
MessageMgtAbstract
com.ebsolutions.soacore.process.messages
showMessage(String,String,int):int
showMessage(String,String,int,boolean,boolean):int
showMessage(List<String>,String,int,boolean,boolean):int
forceInteractiveMessage(String,String,int,boolean,boolean):int
get_messages():String[]
get_append_to_log():boolean
get_log_file():File
set_log_file(File):void
set_log_file(String):void
set_append_to_log(boolean):void
writeMessageInLogFile(String):int

Use the showMessage function to display messages.

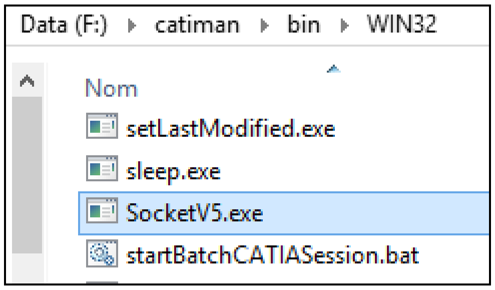
When the getDefaultSilentProcessConfiguration configuration is managed, the **MessageMgtSilent** function is used and showMessage function writes in the log file.

When the getDefaultProcessConfiguration configuration is managed, the **MessageMgtInteractive** function is used and the showMessage function displays in a message box.

The **forceInteractiveMessage** function can be used to force message displays in a dialog box, even when the mode is silent.

Launch Process

The *SocketV5.exe* file, available under the *%TCICV5_DIR%\bin\WIN32* directory, launches processes through socket server management.


Data (F:) ▸ catiman ▸ bin ▸ WIN32

Nom
setLastModified.exe
sleep.exe
SocketV5.exe
startBatchCATIASession.bat

Three parameters must be set:

1. The name of the process including class path and package. For example, *com.MyNewProcess.*

2. A tag (0 or 1) to specify if the process is a blocking process.

   a. **0** if SocketV5.exe is not waiting for the end of com.MyNewProcess before continuing.

   b. **1** if SocketV5.exe is waiting for the end of MyNewProcess before continuing.

3. RACLess server socket port, defined in the *%TCICV5_DIR%\env\tcic_var_env.xml* file.

```
<TcIC_socket_port>
11400
</TcIC_socket_port>
```

For example: :

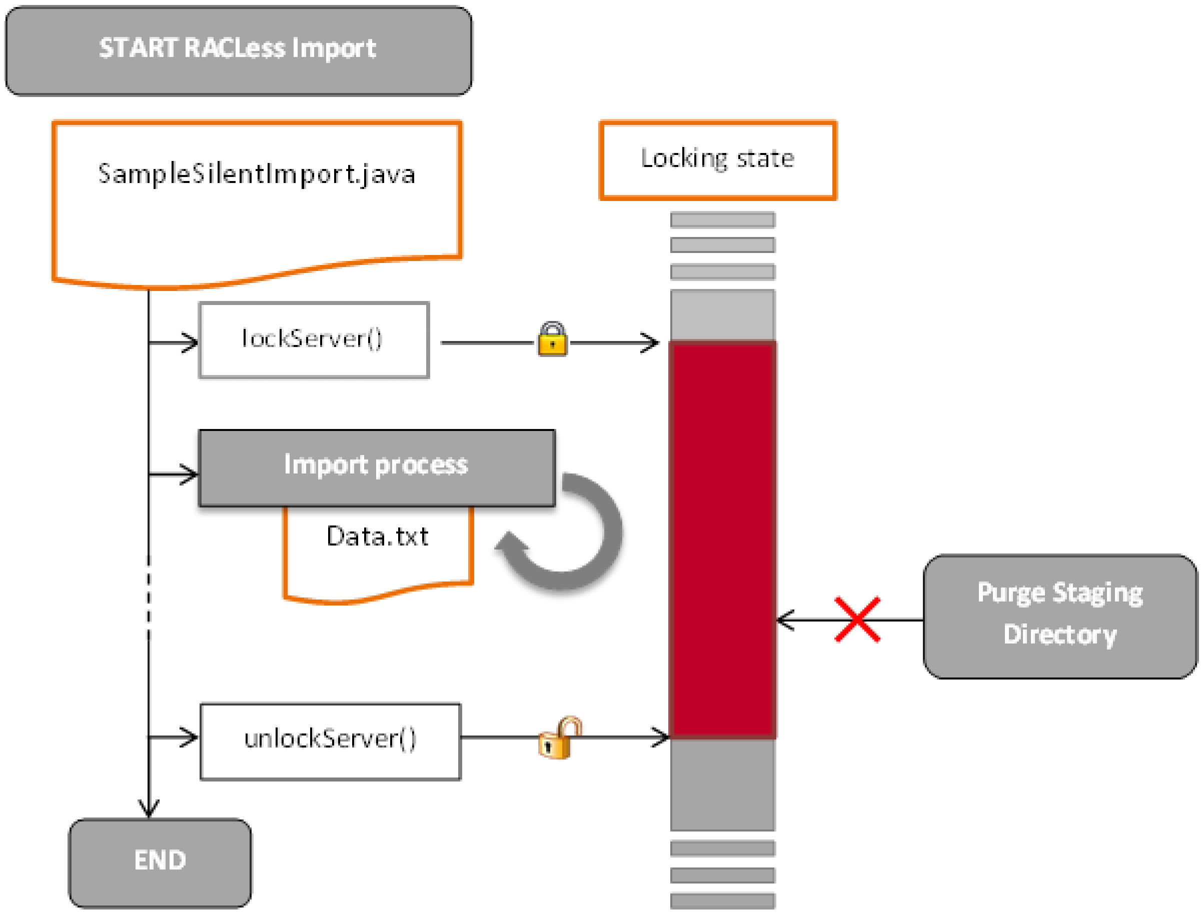
%TCICV5_DIR%\bin\WIN32\socketV5.exe com.sample.SampleSilentImport 0 11400

7.2.4.1.3 Process Lock

Lock RACLess processes so that if one process is running, another process cannot be launched until the first one finishes.

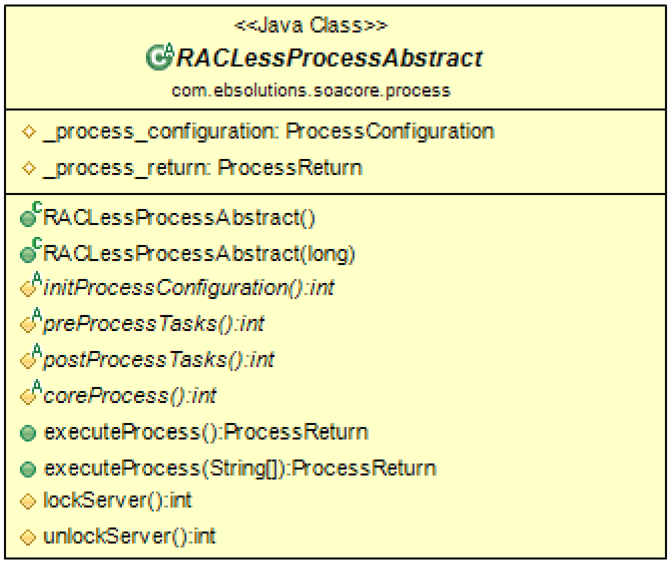
Lock management is also available as a customized implementation. Locking the socket server is **mandatory** to avoid unexpected and incorrect behavior in the sample process.

The example below illustrates what happens when an import RACLess process is currently running and a Purge command launches.



Through the RACLessProcessAbstract class, it is possible to:

- Lock a process using the **lockServer()** method. If the method return a value other than 0, then another process is running.
- Unlock a process using the **unlockServer()** method.


<<Java Class>>
RACLessProcessAbstract
com.ebsolutions.soacore.process
_process_configuration: ProcessConfiguration
_process_return: ProcessReturn
RACLessProcessAbstract()
RACLessProcessAbstract(long)
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int
executeProcess():ProcessReturn
executeProcess(String[]):ProcessReturn
lockServer():int
unlockServer():int

7.2.4.1.4 Selection Management

There are two possibilities for setting selection during a process :

1. tcic_selection.xml file
2. ProcessConfiguration class

Rules

When the selection has been defined by code through ProcessConfiguration, the *tcic_selection.xml* file is not read.

The user is responsible for releasing the selection at the end of the process.

The tcic_selection.xml file

The *tcic_selection.xml* file must be created under *%WORK_DIR%\tmp* and adhere to the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component uid="[…]" bomwindow="[…]"/>
<component …/>
</selection>
```

Description

The `<selection>…</selection>` tag contains a list of Teamcenter Objects (Item, Item Revision, Datasets, BOM Lines, etc.)

Each object is identified by a `<component>` tag and a `uid` attribute.   Each object under the `<component>` tag is managed as a ModelObject.

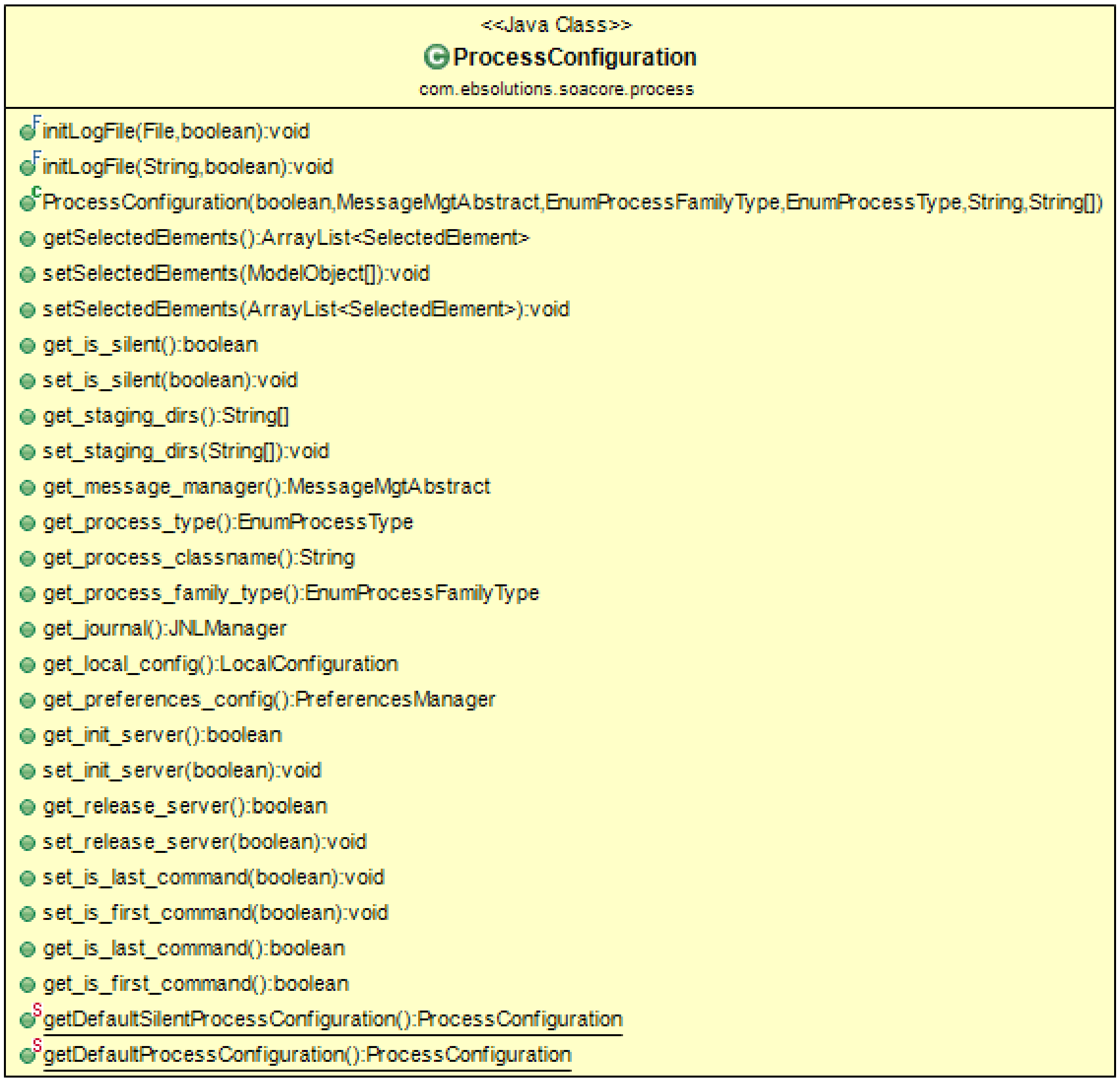
For a BOM Line object, the attribute, `bomwindow`, must be specified.

For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component uid="QPRJPwqqohVQuA"/>
<component uid="QjbJPwqqohVQuA"/>
<component uid="BOM::40133" bomwindow="BOM::40128"/>
<component uid="BOM::40135" bomwindow="BOM::40128"/>
<component uid="BOM::40137" bomwindow="BOM::40128"/>
<component uid="BOM::40148" bomwindow="BOM::40164"/>
</selection>
```

The example above indicates, two objects of an Item or ItemRevision or Dataset type, three BOM lines originating from the same BOM Window (`BOM::40128`) and one BOM line originating from a specific BOM Window (`BOM::40164`)

ProcessConfiguration



```
<<Java Class>>
ProcessConfiguration
com.ebsolutions.soacore.process

initLogFile(File,boolean):void
initLogFile(String,boolean):void
ProcessConfiguration(boolean,MessageMgtAbstract,EnumProcessFamilyType,EnumProcessType,String,String[])
getSelectedElements():ArrayList<SelectedElement>
setSelectedElements(ModelObject[]):void
setSelectedElements(ArrayList<SelectedElement>):void
get_is_silent():boolean
set_is_silent(boolean):void
get_staging_dirs():String[]
set_staging_dirs(String[]):void
get_message_manager():MessageMgtAbstract
get_process_type():EnumProcessType
get_process_classname():String
get_process_family_type():EnumProcessFamilyType
get_journal():JNLManager
get_local_config():LocalConfiguration
get_preferences_config():PreferencesManager
get_init_server():boolean
set_init_server(boolean):void
get_release_server():boolean
set_release_server(boolean):void
set_is_last_command(boolean):void
set_is_first_command(boolean):void
get_is_last_command():boolean
get_is_first_command():boolean
getDefaultSilentProcessConfiguration():ProcessConfiguration
getDefaultProcessConfiguration():ProcessConfiguration
```

The **setSelectedElements** function defines the list of the selected ModelObject. It is recommended to define this list early in the process; for example in the **initProcessConfiguration** or **preProcessTasks** function of your process.

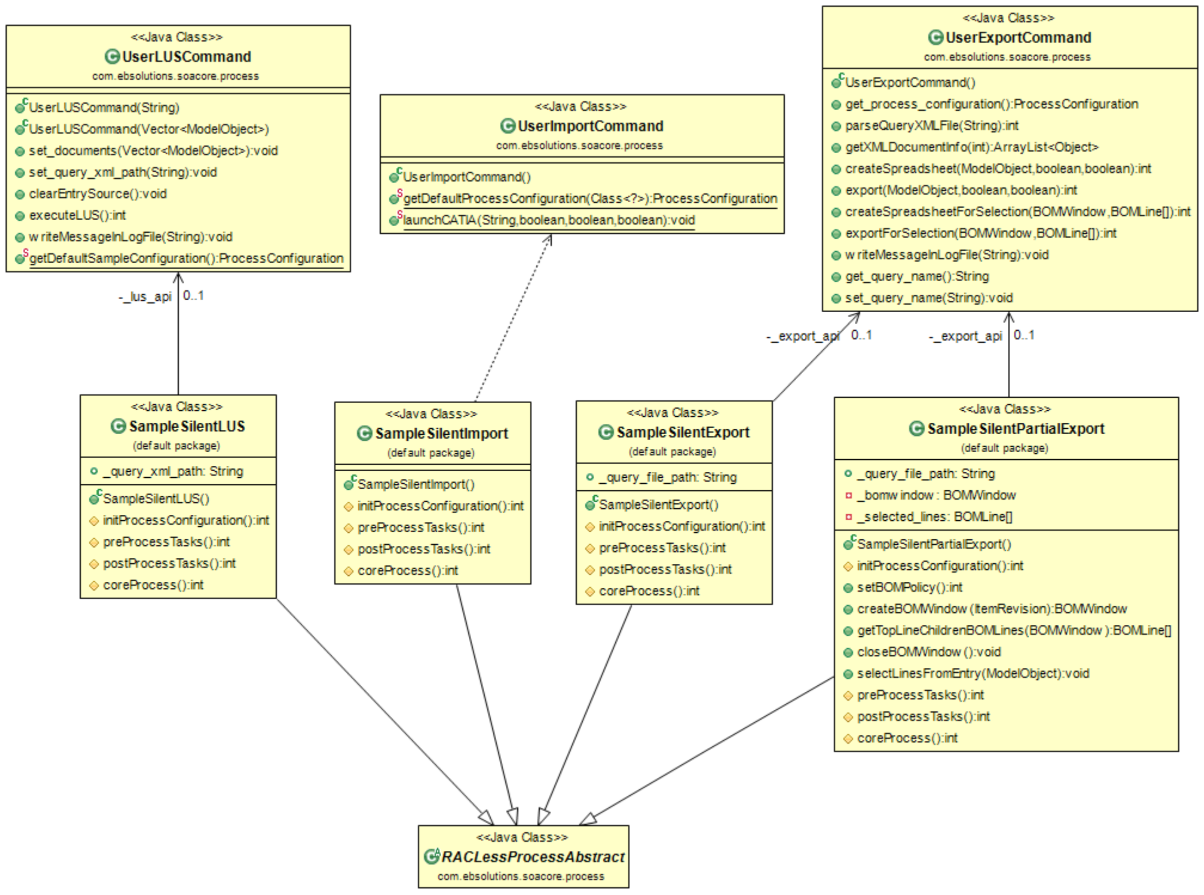
The **clearSelectedElements** function clears the selection when needed.

7.2.4.1.5 *Silent Processes with JAVA Code Customization*

Similar to each RACLess process, silent processes also inherit from the RACLessProcessAbstract class.

All RACLess processes are silent processes beginning with Teamcenter Integration for CATIA V5 (10.0.0)

Available processes and their APIs

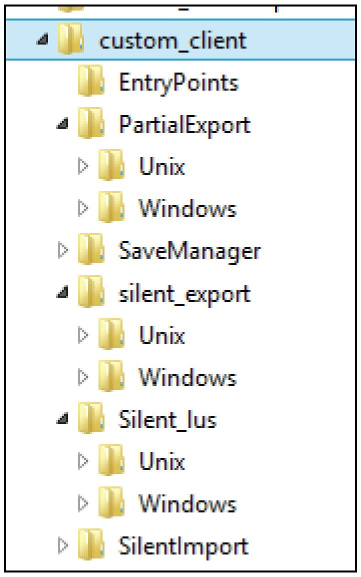

<<Java Class>>
UserLUSCommand
com.ebsolutions.soacore.process
UserLUSCommand(String)
UserLUSCommand(Vector<ModelObject>)
set_documents(Vector<ModelObject>):void
set_query_xml_path(String):void
clearEntrySource():void
executeLUS():int
writeMessageInLogFile(String):void
getDefaultSampleConfiguration():ProcessConfiguration
<<Java Class>>
UserImportCommand
com.ebsolutions.soacore.process
UserImportCommand()
getDefaultProcessConfiguration(Class<?>):ProcessConfiguration
launchCATIA(String,boolean,boolean,boolean):void
<<Java Class>>
UserExportCommand
com.ebsolutions.soacore.process
UserExportCommand()
get_process_configuration():ProcessConfiguration
parseQueryXMLFile(String):int
getXMLDocumentInfo(int):ArrayList<Object>
createSpreadsheet(ModelObject,boolean,boolean):int
export(ModelObject,boolean,boolean):int
createSpreadsheetForSelection(BOMWindow,BOMLine[]):int
exportForSelection(BOMWindow,BOMLine[]):int
writeMessageInLogFile(String):void
get_query_name():String
set_query_name(String):void
-_lus_api
0..1
-_export_api
0..1
-_export_api
0..1
<<Java Class>>
SampleSilentLUS
(default package)
_query_xml_path: String
SampleSilentLUS()
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int
<<Java Class>>
SampleSilentImport
(default package)
SampleSilentImport()
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int
<<Java Class>>
SampleSilentExport
(default package)
_query_file_path: String
SampleSilentExport()
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int
<<Java Class>>
SampleSilentPartialExport
(default package)
_query_file_path: String
_bomwindow : BOMWindow
_selected_lines: BOMLine[]
SampleSilentPartialExport()
initProcessConfiguration():int
setBOMPolicy():int
createBOMWindow(ItemRevision):BOMWindow
getTopLineChildrenBOMLines(BOMWindow):BOMLine[]
closeBOMWindow():void
selectLinesFromEntry(ModelObject):void
preProcessTasks():int
postProcessTasks():int
coreProcess():int
<<Java Class>>
RACLessProcessAbstract
com.ebsolutions.soacore.process

UserExportCommand : APIs for silent export.

UserLUSCommand : APIs for silent Load, Update, Save.

UserImportCommand : APIs for silent import.

Samples are available under *%TCICV5_DIR%\custom_client.*



7.2.4.1.6 The tcic_selection.xml File

The **tcic_selection.xml** file defines the selected BOM lines, the selected Drawing dataset, the selected item revision or the BOM Window.

The *tcic_selection.xml* file is used to define the configuration used during BOM Window creation. This BOM Window corresponds to the item revision UID defined at the top line of the file and reflects the selected BOM lines in Teamcenter corresponding to the occurrences UID path defined in the file.

Three types of components are supported:

- *itemrevision* corresponding to an item revision selected in My Teamcenter or Classification.
- *dataset* corresponding to a dataset selected in My Teamcenter or Classification.
- *bomwindow* corresponding to a BOM window selected in the Structure Manager or Design Context.
    - The selected BOM lines are listed under the bomwindow component, under `<bomline>` tags. When no tag is defined for the BOM line, the process is executed on the top line.

The file should be created under *%WORK_DIR%\tmp* and adhere according the following format. The sample shown below can be found in the online documentation.

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component type="itemrevision" uid="www"/>
<component type="dataset" uid="xxx"/>
<component type="bomwindow" uid="yyy">
    <bomline uid="zzz"/>
</component>
<component type="itemrevision">
    <item item_id="00012" object_type="ICATProduct"
object_name="Product1" prop1="124"/>
    <revision item_revision_id="A"/>
 </component>
<component type="bomwindow">
    <item item_id="00012" object_type="ICATProduct"
object_name="Product1" prop1="124"/>
    <revision item_revision_id="A"/>
    <configuration>
          <revrule name="mmm"/>
          <effectivity date="qqq" unit="rrr" item="sss"/>
          <optionset name="ttt"/>
          <variantrule name="uuu"/>
    </configuration>
    <bomline path="aaa/bbb/ccc/ddd"/>
</component>
</selection>
```

The following example illustrates 3 BOM Windows.

- A selected BOM line with a configured revision rule.
- No BOM Line selection, so the process is launched on the top line with an option set configured.
- Two selected BOM Lines.

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component type="bomwindow">
    <item item_id="A1_item_id" object_type="A1_object_type" />
    <revision item_revision_id="A1_item_revision_id" />
    <configuration>
          <revrule name="Working; Any Status" />
    </configuration>
    <bomline path="A2_occ_uid/A3_occ_uid" />
</component>
<component type="bomwindow">
    <item item_id="A10_item_id" object_type="A10_object_type" />
    <revision item_revision_id="A10_item_revision_id" />
    <configuration>
          <optionset name="color_blue"/>
    </configuration>
</component>
<component type="bomwindow">
    <item item_id="A100_item_id" object_type="A100_object_type" />
    <revision item_revision_id="A100_item_revision_id" />
    <bomline path="A200_occ_uid/P100_occ_uid" />
    <bomline path="A200_occ_uid/A300_occ_uid/P200_occ_uid" />
</component>
</selection>
```

7.2.4.1.7 User Interface Implementations: Main Classes

Two interfaces, ICatiaUserExit and ICommandEvent, are available to customize RACLess processes.

ICatiaUserExit

The ICatiaUserExit interface defines five functions called in the following processes.

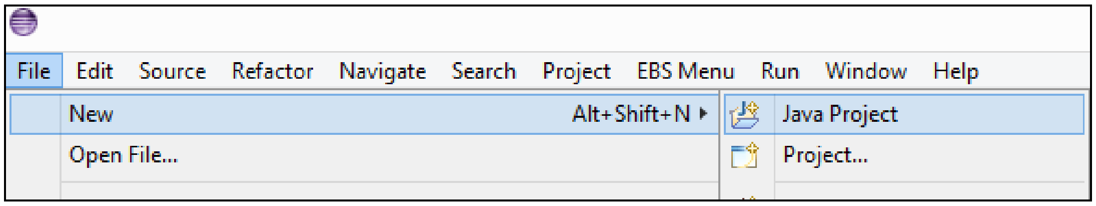
Function	TcIC RACLess Processes

userExitAfterAllExportedFiles(…)
userExitAtEndOfLoadProcesses(…)
userExitAfterAllExportedFiles(…)
	Create Export Spreadsheet
	Export
	Background Round Robin

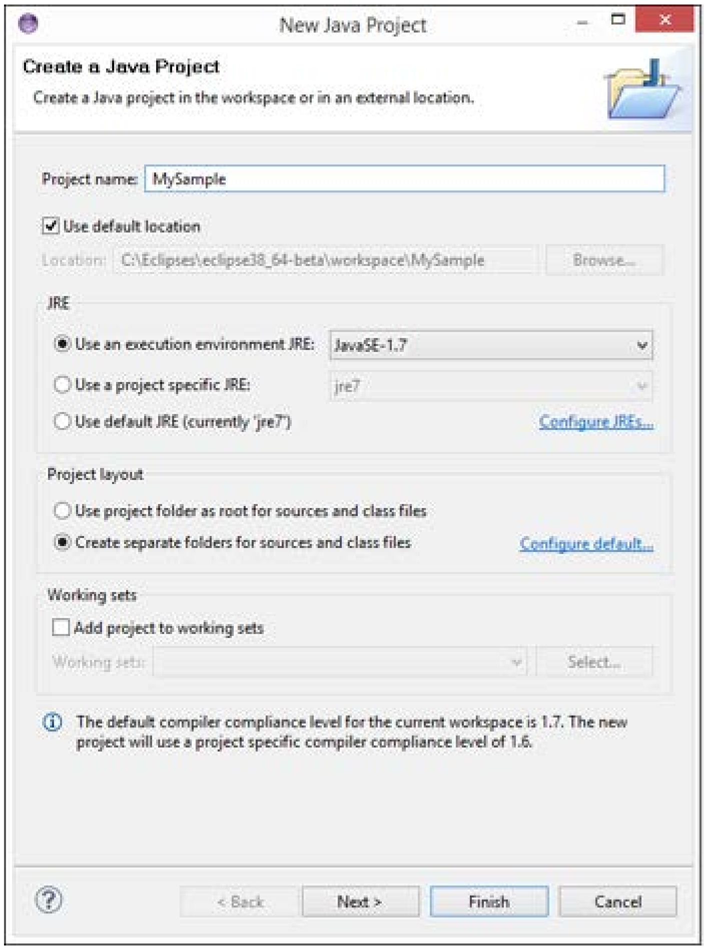
userExitAfterAllImportedFiles(…)	Import Background Round Robin
userExitModifyDesc(…)	Create Export Spreadsheet Export Background Round Robin Import

Implementation Example

1. In Eclipse, create a new Java project.



2. Set the **Project name** then select the compatible JRE version: **Java 1.7** for Teamcenter 10.1.5 and Teamcenter 11.2.1. Pick **Next** to continue.


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.

Project name: MySample

☑ Use default location

Location: C:\Eclipses\eclipse38_64-beta\workspace\MySample

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: jre7

○ Use default JRE (currently 'jre7')

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

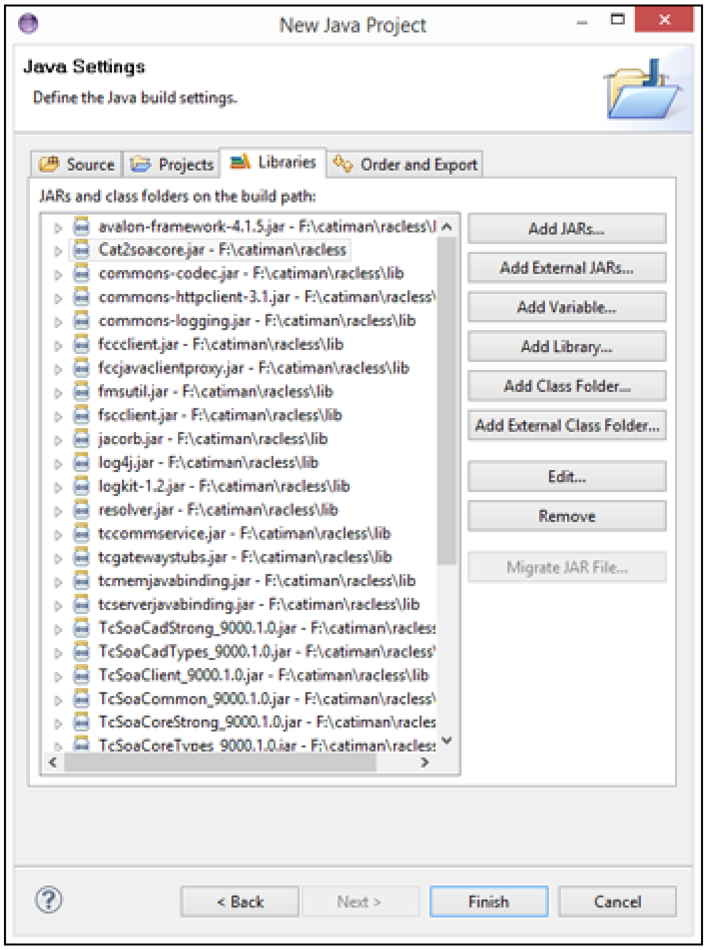
☐ Add project to working sets

The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.6.

3. Select the **Libraries** tab, then select the following libraries required for the project. Pick **Finish** when done.

   a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

b. Cat2soacore.jar library contained in *%TCICV5_DIR%\racless.*

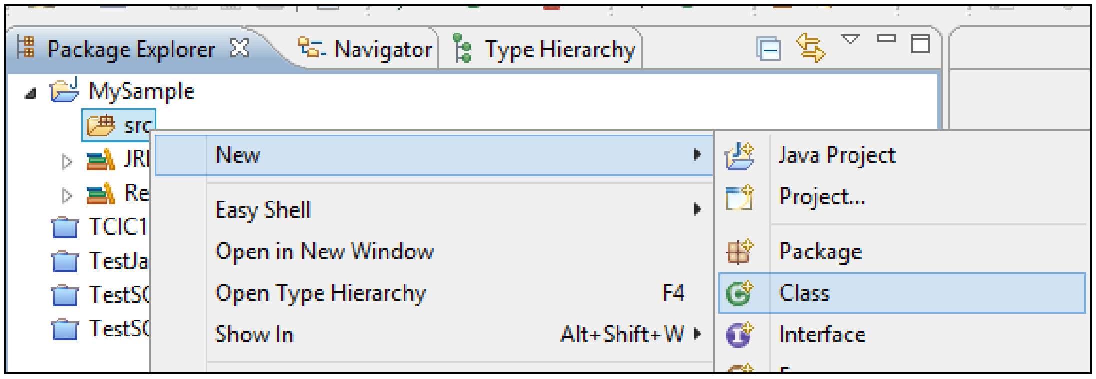


JARs and class folders on the build path:

```
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless
```

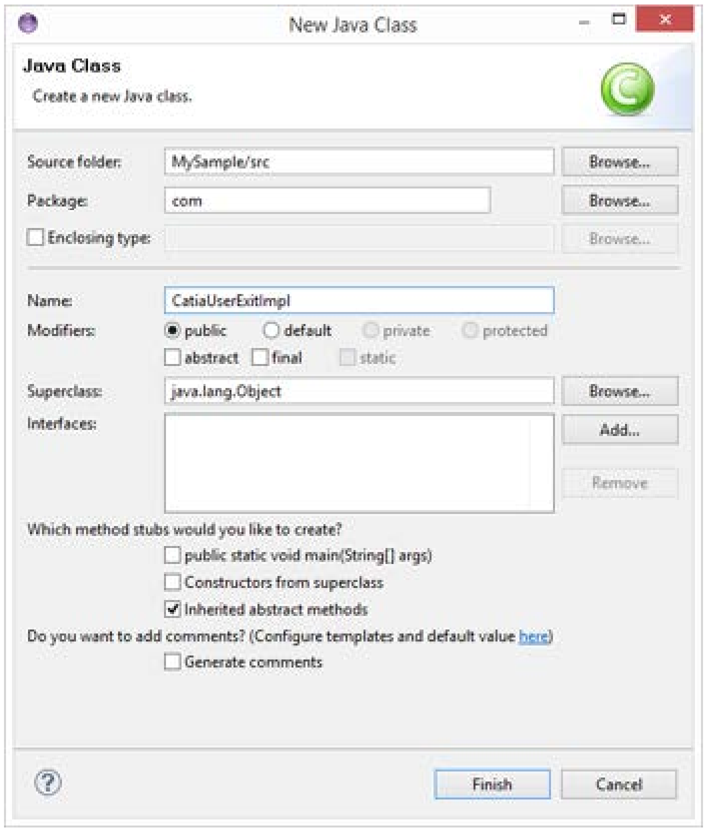
Create the ICatiaUserExit API Implementation

4. Select the **src** folder, right-click and then select **New >> Class**.



5. Complete the **Package:** and **Name:** fields on the New Java Class window.

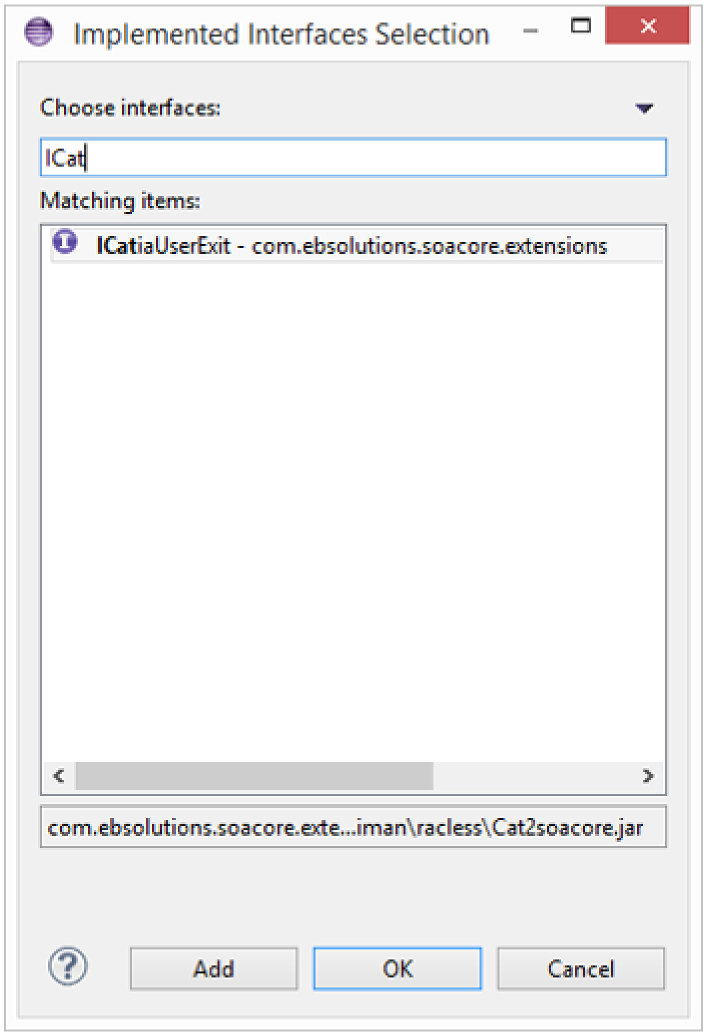
6. Untoggle **public static void main(String[] args)** and toggle the **Inherited abstract methods** option on. Pick the **Add** button next to the Interfaces: field.



Java Class
Create a new Java class.

Source folder: MySample/src
Package: com
[ ] Enclosing type:
Name: CatiaUserExitImpl
Modifiers: [✓ public] default private protected
[ ] abstract [ ] final [ ] static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
[ ] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

7. Enter *ICatiaUserExit* in the Choose interfaces: field, select **ICatiaUserExit** from the Matching Items box, then pick **OK**.

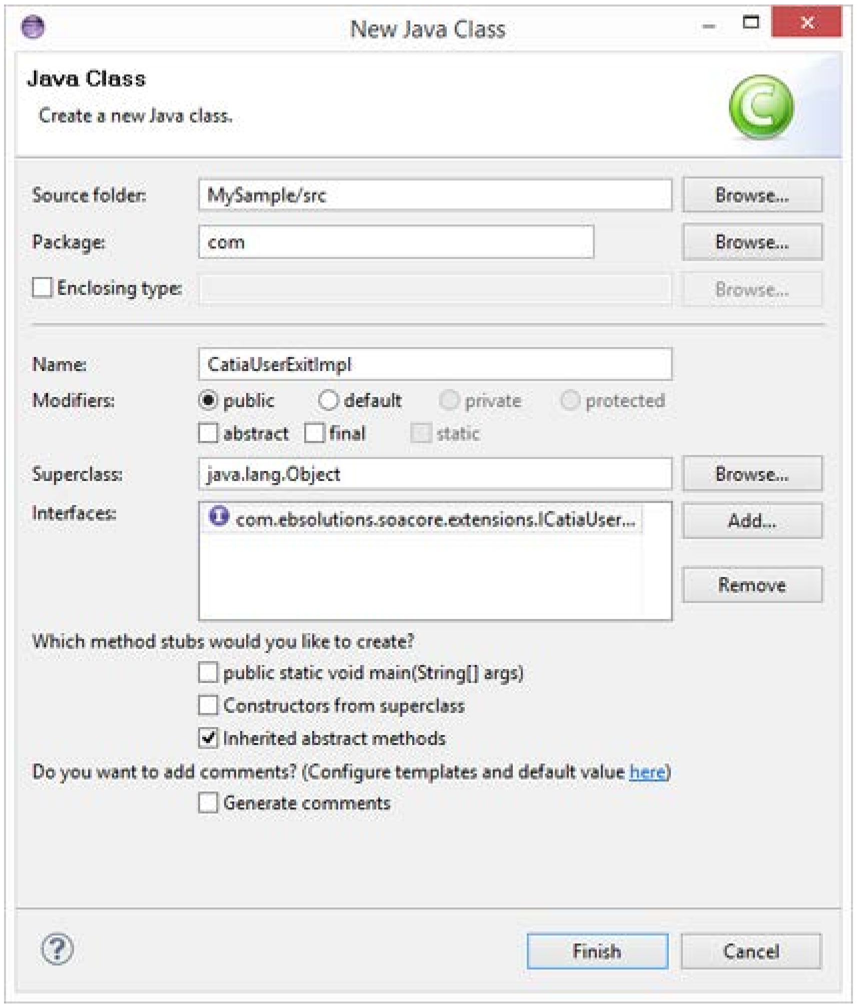


ICat

ICatiaUserExit - com.ebsolutions.soacore.extensions

com.ebsolutions.soacore.exte...iman\racless\Cat2soacore.jar

8. Pick **Finish**.



Source folder: MySample/src
Package: com
☐ Enclosing type:
Name: CatiaUserExitImpl
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces: com.ebsolutions.soacore.extensions.ICatiaUser...
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

As a result ICatiaUserExit is automatically written as:

```
<code>
public class CatiaUserExitImpl implements ICatiaUserExit {

    @Override
    public void userExitAfterAllExportedFiles(Vector<String> arg0)
                   throws IOException {
           // TODO Auto-generated method stub

    }

    @Override
    public int userExitAfterAllExportedFiles(Connection arg0,
                   Vector<String> arg1, Vector<String> arg2,
Vector<String> arg3,
                   Vector<String> arg4) {
           // TODO Auto-generated method stub
```

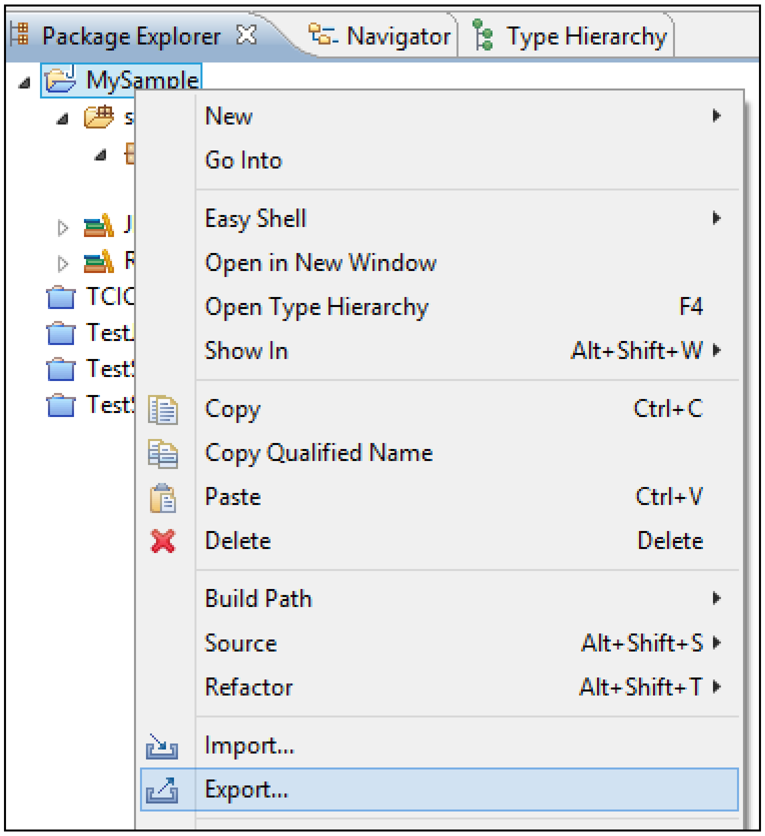
```
            return 0;
     }

     @Override
     public int userExitAfterAllImportedFiles(Connection arg0,
                    Vector<String> arg1, Vector<String> arg2,
Vector<String> arg3,
                    Vector<String> arg4) {
            // TODO Auto-generated method stub
            return 0;
      }

      @Override
      public void userExitAtEndOfLoadProcesses() throws IOException
{
            // TODO Auto-generated method stub
      }

      @Override
      public void userExitModifyDesc(String arg0) {
            // TODO Auto-generated method stub

      }

}
</code>
```

9. Modify the userExitAfterAllImportedFiles and userExitModifyDesc functions.

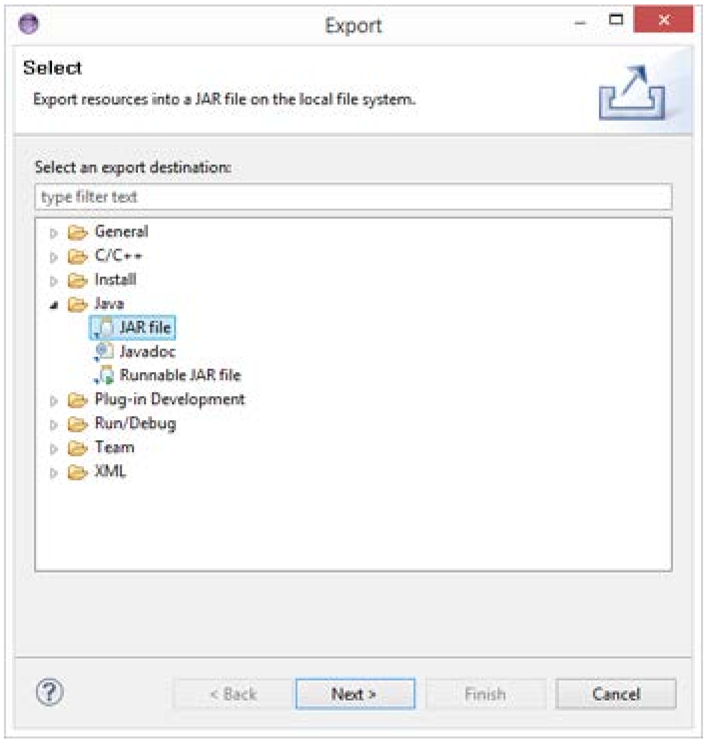
```
<code>
     @Override
     public int userExitAfterAllImportedFiles(Connection
i_connection,
          Vector<String>  i_file_name_list,
          Vector<String>  i_item_id_list,
          Vector<String>  i_item_rev_list,
          Vector<String>  i_dataset_uid_list)
     {
             for (String str : i_file_name_list)
             {
                    System.out.println("i_file_name_list->" + str);
             }
             for (String str : i_item_id_list)
             {
                    System.out.println("i_item_id_list->" + str);
             }
             for (String str : i_item_rev_list)
             {
                    System.out.println("i_item_rev_list->" + str);
             }
             return 0;
}

@Override
public void userExitModifyDesc(String arg0) {
      System.out.println("userExitModifyDesc arg=" + arg0);
}
</code>
```

Generate a Library

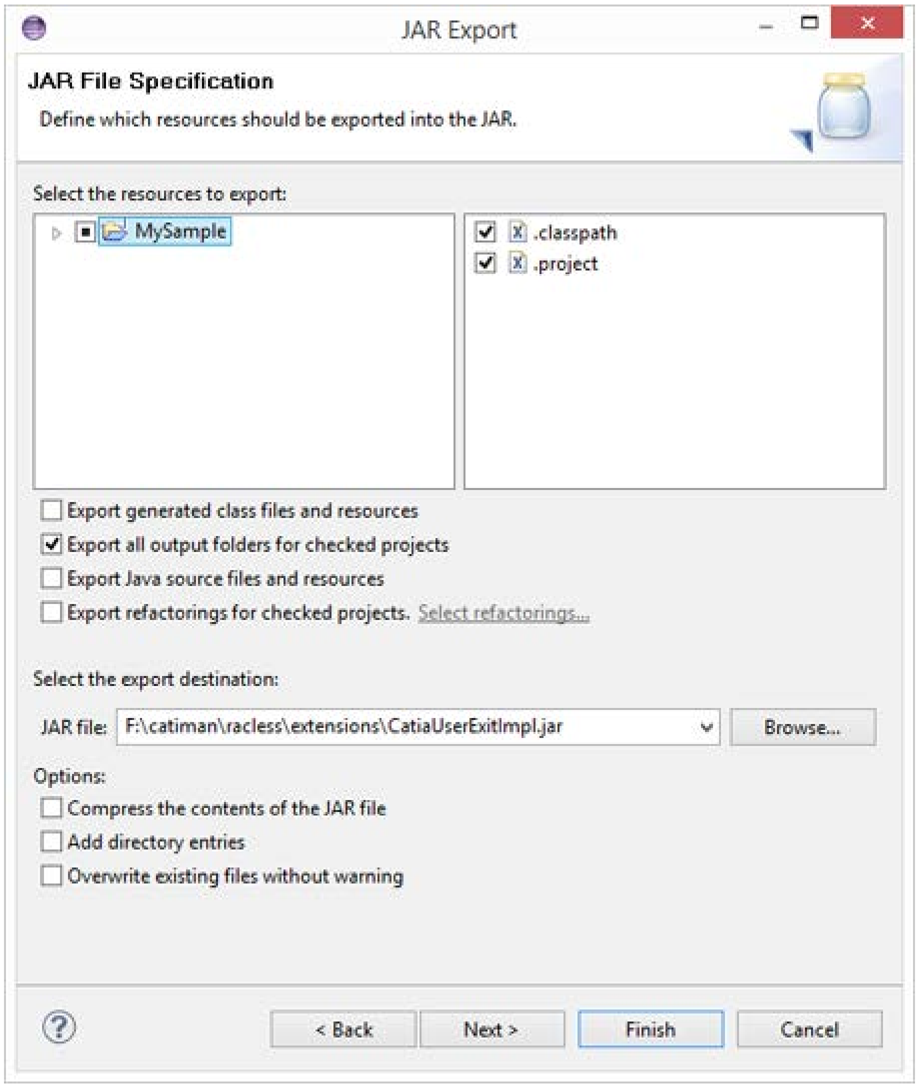
10. Highlight the project folder, then right-click and pick **Export**.



11. Toggle the **Java** folder open, select *JAR file*, then pick **Next**.



12. Select the project then enter a name and library location in the **JAR file** field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

[■] MySample

[✓] .classpath
[✓] .project

[ ] Export generated class files and resources
[✓] Export all output folders for checked projects
[ ] Export Java source files and resources
[ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

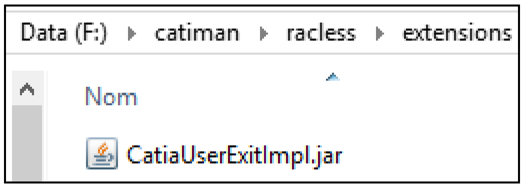
JAR file: F:\catiman\racless\extensions\CatiaUserExitImpl.jar

Options:

[ ] Compress the contents of the JAR file
[ ] Add directory entries
[ ] Overwrite existing files without warning

Result:

The user library is created and automatically loads during RACLess processes. It is not necessary to update the classpath.

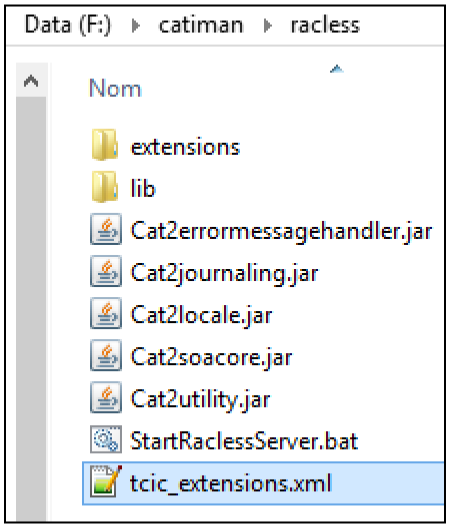

Data (F:) ▸ catiman ▸ racless ▸ extensions

Nom

CatiaUserExitImpl.jar

Define additional user entry point implementations

13. Edit the *tcic_extensions.xml* file within the *%TCICV5_DIR%\racless* directory to define additional user entry point implementations.


Data (F:) ▸ catiman ▸ racless

Nom

extensions
lib
Cat2errormessagehandler.jar
Cat2journaling.jar
Cat2locale.jar
Cat2soacore.jar
Cat2utility.jar
StartRaclessServer.bat
tcic_extensions.xml

Each implementation is referenced under a specific `<entry>` tag; `id` corresponds to the interface that has been implemented, `handler` corresponds to the implementation definition (its package and class) and `filename` corresponds the name of the user library.

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
     <entry
       id="com.ebsolutions.soacore.extensions.ICatiaUserExit"
       filename="CatiaUserExitImpl.jar"
       handler="com.CatiaUserExitImpl">
     </entry>
</extensions>
```

Several `<entry>` tags for the same interface may be defined, but the implementations must be different. For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
     <entry
       id="com.ebsolutions.soacore.extensions.ICatiaUserExit"
       filename="CatiaUserExitImpl.jar"
       handler="com.CatiaUserExitImpl">
     </entry>
     <entry
       id="com.ebsolutions.soacore.extensions.ICatiaUserExit"
       filename="sample2.jar"
       handler="mysample.MyImpl2">
     </entry>
     <entry
       id="com.ebsolutions.soacore.process.ICommandEvent"
       filename="CommandEventImpl.jar"
```

```
        handler="com.CommandEventImpl">
      </entry>
</extensions>
```

Each implementation is called by the RACLess processes which call this interface.

Refer to the Import RACLess section for more details related to specific implementations.

In the next example, traces are written to the *tcic_racless_out_<timestamp>.log* file under the *%TEMP%* directory. Traces related to userExitModifyDesc calls are shown in red and traces related to userExitAfterAllImportedFiles calls are shown in blue after files are imported.

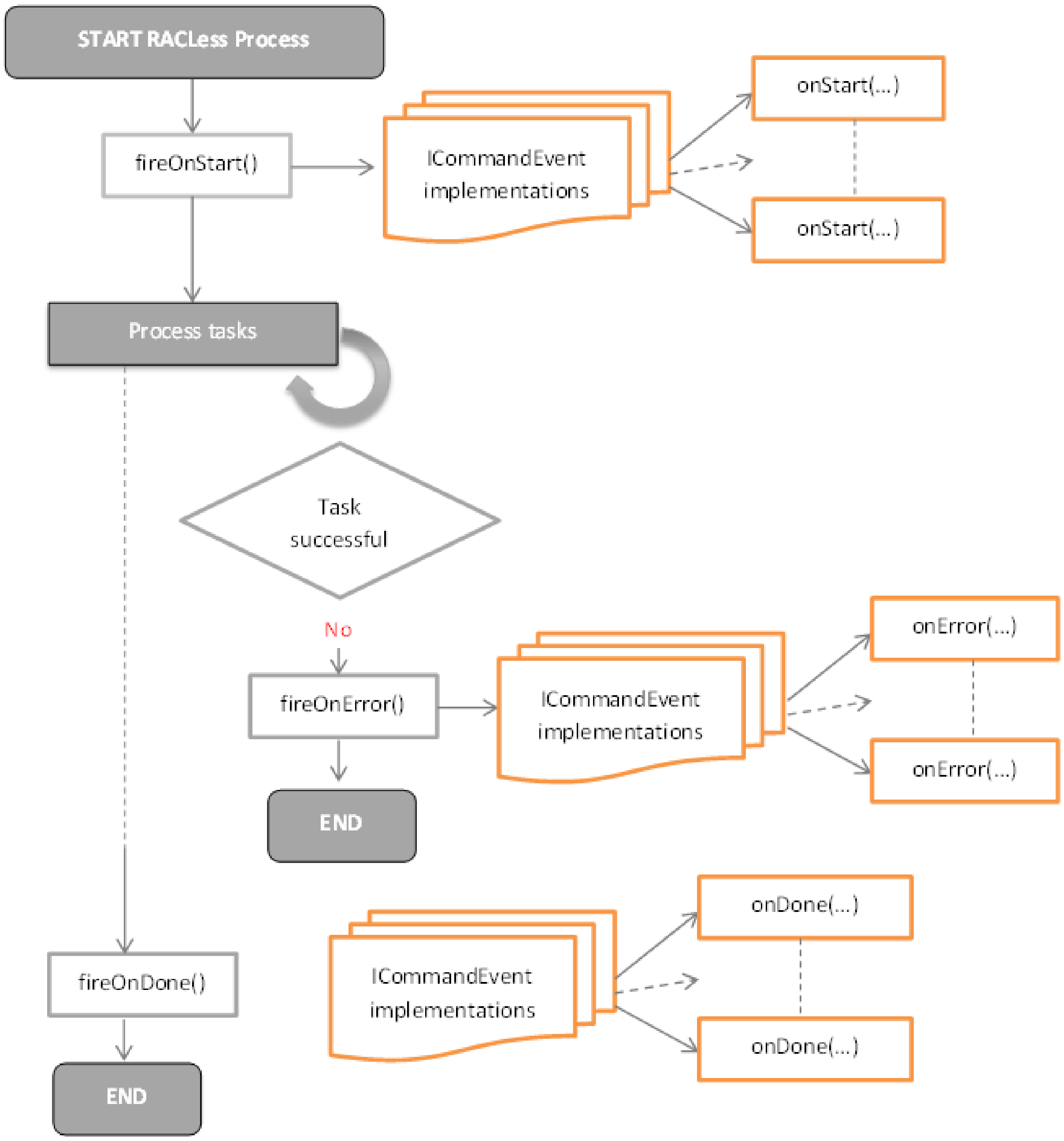
```
 […]
TcICSOAServices.getTabStr(result_file)
userExitModifyDesc arg=SAVE1
send response [SEMAPH]
Command sent [SaveFilesCommand]
Command dispatched [SaveFilesCommand]
callRACLessCommand com.ebsolutions.soacore.process.SaveFilesCommand
ICommandEvent : Interface not implemented
TcICSOAServices.getTabStr(result_file)
TcICSOAServices.getTabStr(import_log_file)
i_file_name_list->F:\catiman_tmp\catuii\050372_A.CATProduct
i_item_id_list->050372
i_item_rev_list->A
userExitModifyDesc arg=SAVE2
send response [SEMAPH]
[…]
```

ICommandEvent

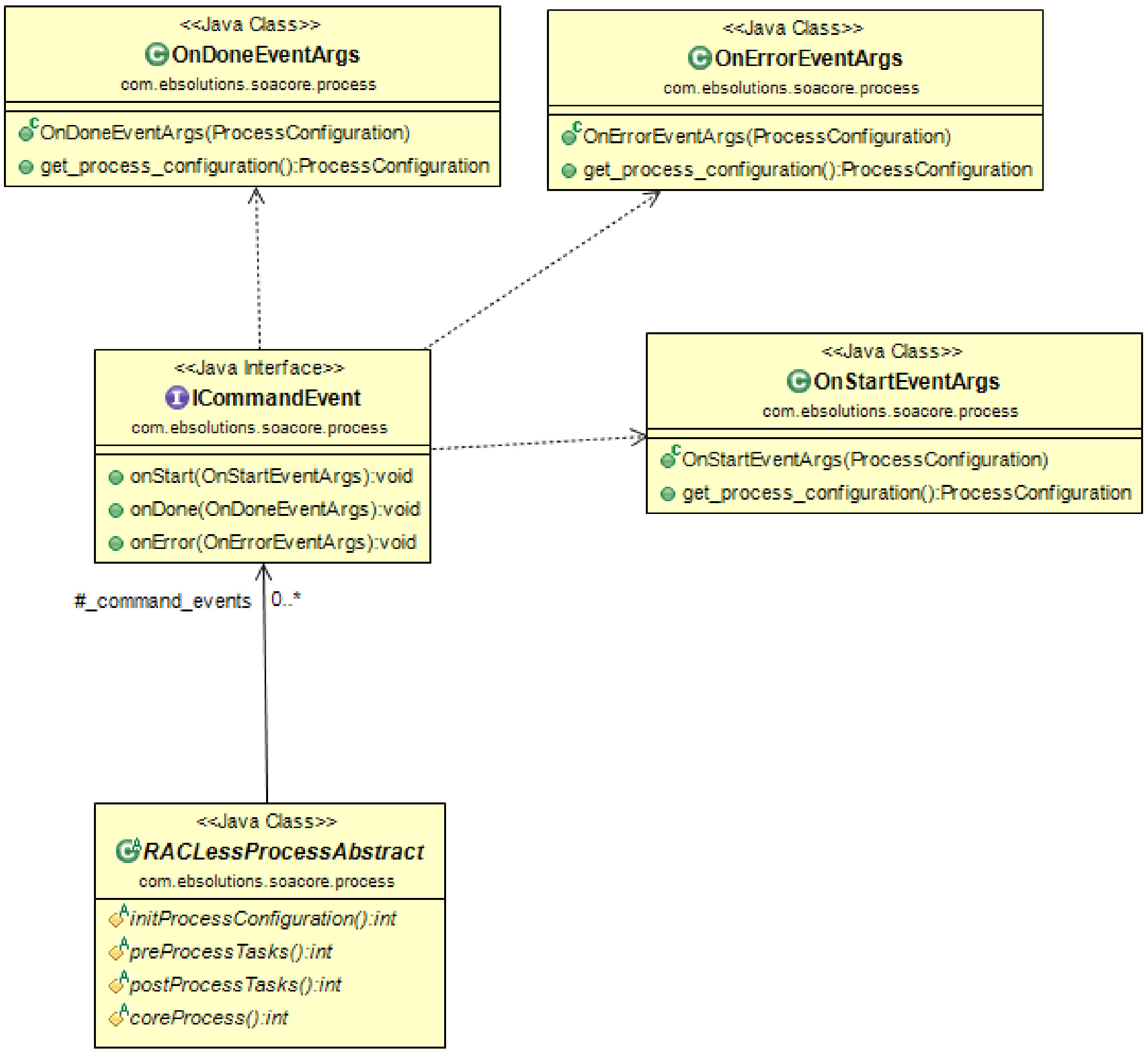
Some events are generated during RACLess processes.

- OnStart(): generates at the beginning of RACLess processes.
- OnError(): generates if an error occurred during the process.
- OnDone():generates at the end of RACLess processes, if no error occurred.

These events are defined within the ICommandEvent interface; pre/post actions may be defined for each RACLess processes.



Information passed through several events are managed in the OnDoneEventArgs, OnErrorEventArgs and OnStartEventArgs classes.


<<Java Class>>
OnDoneEventArgs
com.ebsolutions.soacore.process
OnDoneEventArgs(ProcessConfiguration)
get_process_configuration():ProcessConfiguration
<<Java Class>>
OnErrorEventArgs
com.ebsolutions.soacore.process
OnErrorEventArgs(ProcessConfiguration)
get_process_configuration():ProcessConfiguration
<<Java Interface>>
ICommandEvent
com.ebsolutions.soacore.process
onStart(OnStartEventArgs):void
onDone(OnDoneEventArgs):void
onError(OnErrorEventArgs):void
<<Java Class>>
OnStartEventArgs
com.ebsolutions.soacore.process
OnStartEventArgs(ProcessConfiguration)
get_process_configuration():ProcessConfiguration
#_command_events
0..*
<<Java Class>>
RACLessProcessAbstract
com.ebsolutions.soacore.process
initProcessConfiguration():int
preProcessTasks():int
postProcessTasks():int
coreProcess():int

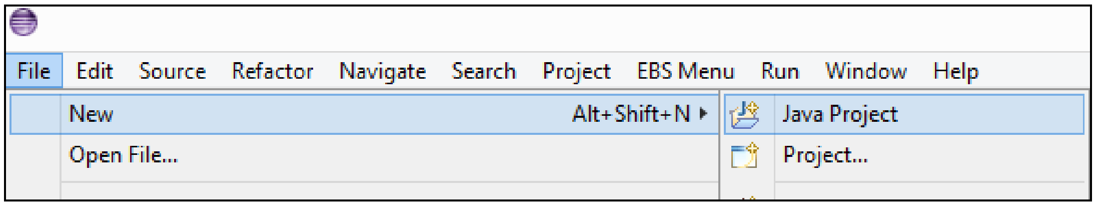
Configuration, error and/or warning messages are available for each event. The sample below shows an onError event:

```
@Override
public void onError(OnErrorEventArgs arg0) {
      ProcessConfiguration config =
arg0.get_process_configuration();
      String[] messages =
config.get_message_manager().get_messages();
      for (String str: messages)
      {
             System.out.println(str);
```

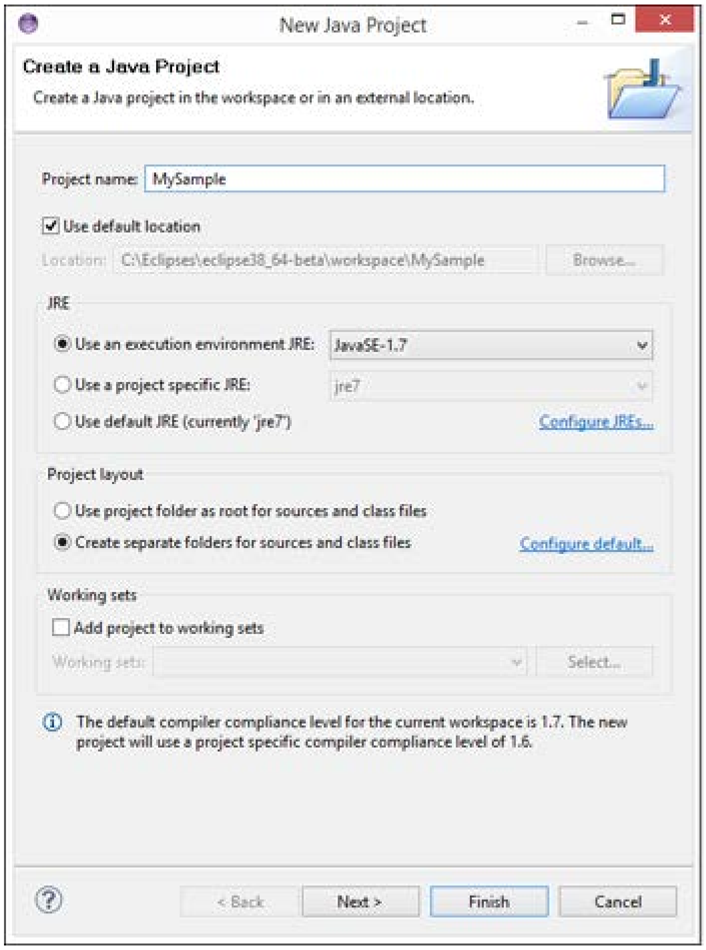
```
        }
}
```

Implementation Example

1. In Eclipse, create a new Java project.



2. Enter the **Project name** then select the compatible JRE version: **Java 1.6** with Teamcenter 9.1.2.9 or **Java 1.7** for Teamcenter 10.1.2. Pick **Next** to continue.


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MySample

☑ Use default location

Location: C:\Eclipses\eclipse38_64-beta\workspace\MySample

JRE

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: jre7

○ Use default JRE (currently 'jre7')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

☐ Add project to working sets

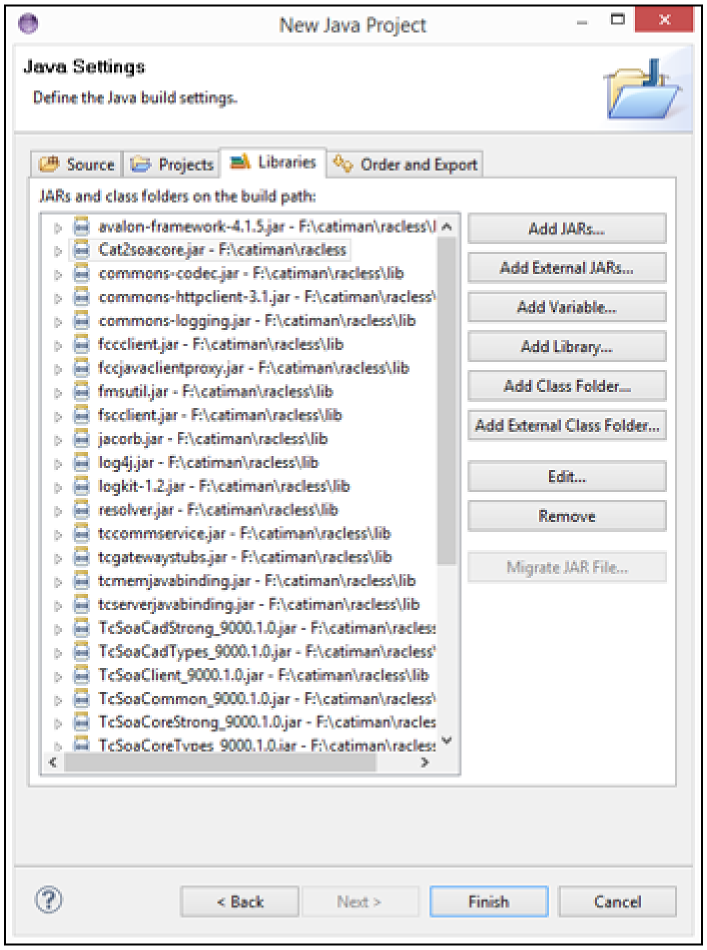
Working sets:

The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.6.

3. Select the **Libraries** tab, then select the following libraries required for the project. Pick **Finish** when done.

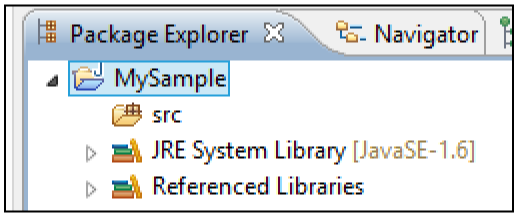
a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

b. Cat2soacore.jar library contained in *%TCICV5_DIR%\racless.*



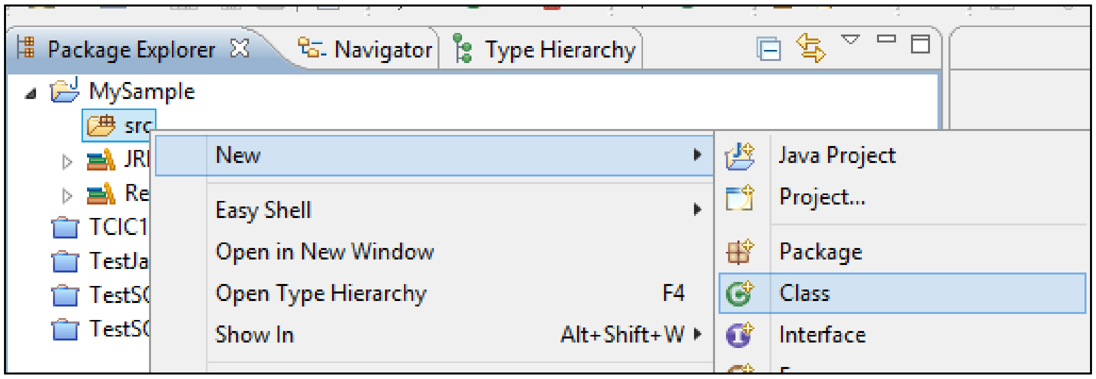
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless

Result:



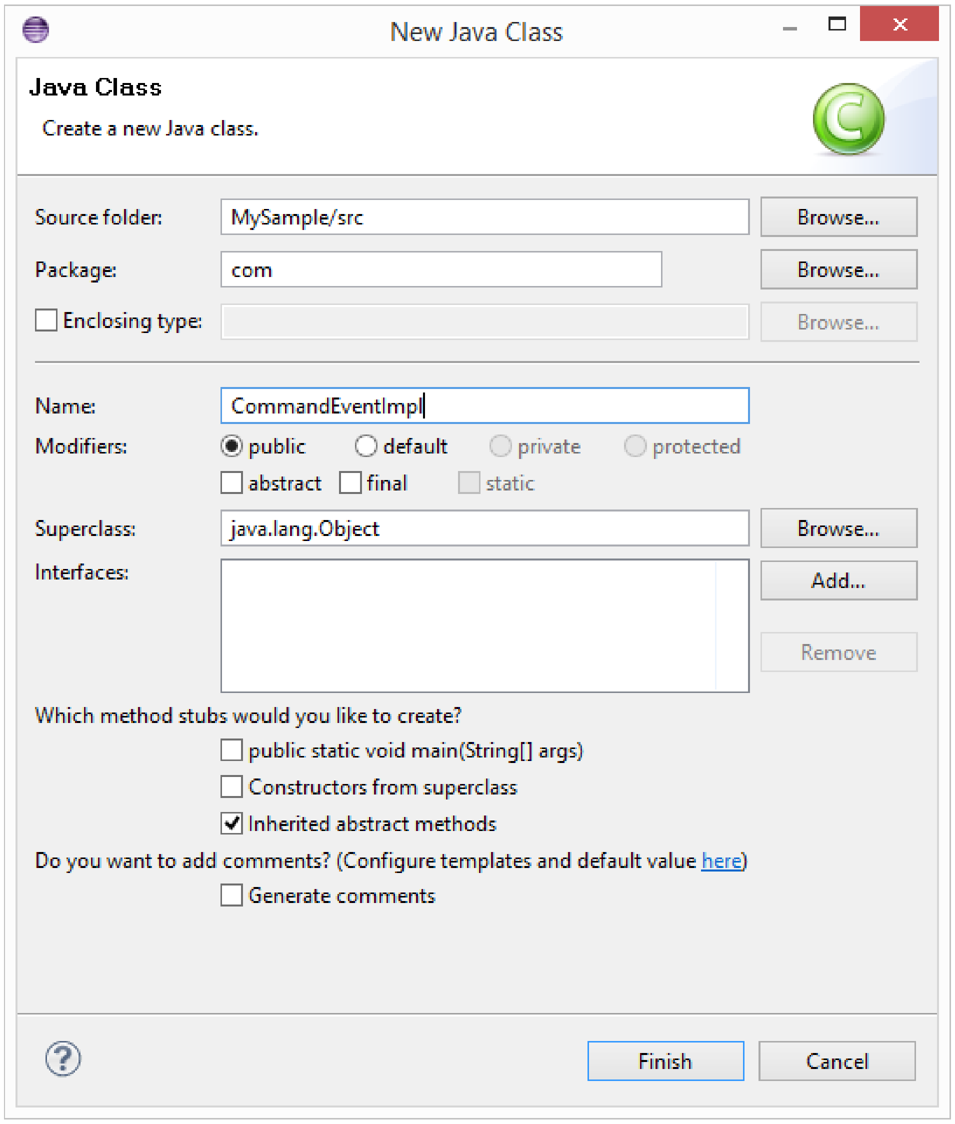
Create the ICommandEvent Interface

4. Select the **src** folder, right-click and then select **New >> Class**.



5. Complete the **Package:** and **Name:** fields on the New Java Class window.

6. Untoggle **public static void main(String[] args)** and toggle **Inherited abstract methods** option. Pick the **Add** button next to the Interfaces: field.



Java Class

Create a new Java class.

Source folder: MySample/src

Package: com

Enclosing type: ☐

Name: CommandEventImpl

Modifiers: ● public ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

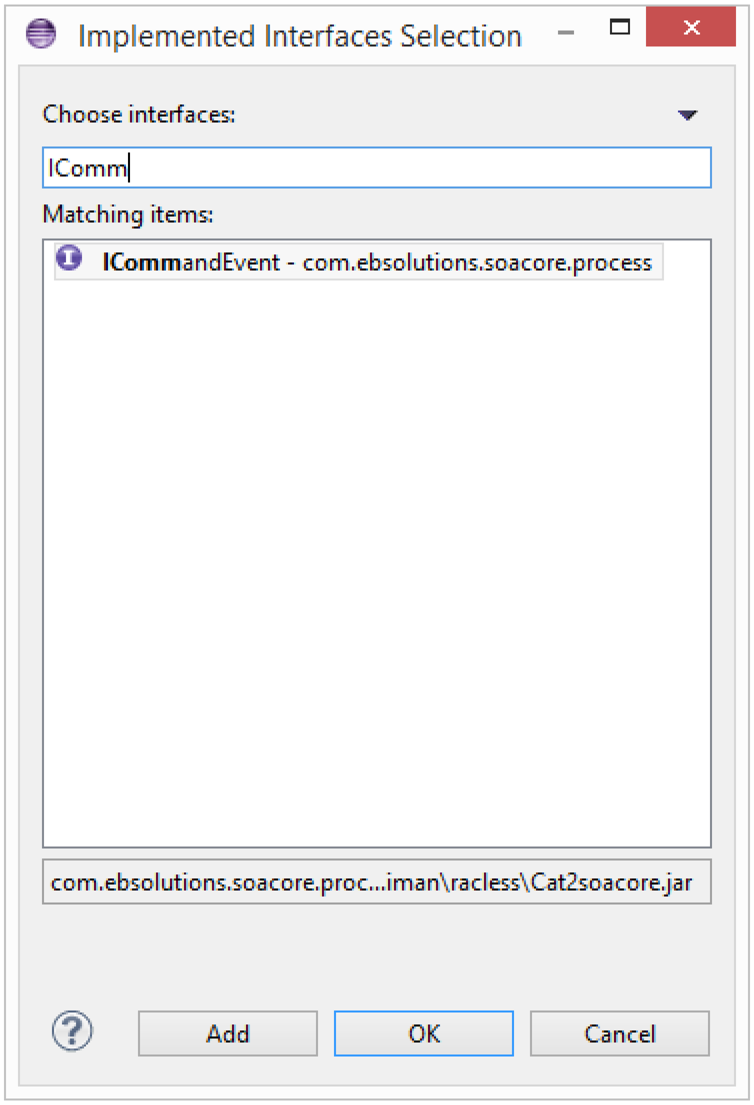
☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

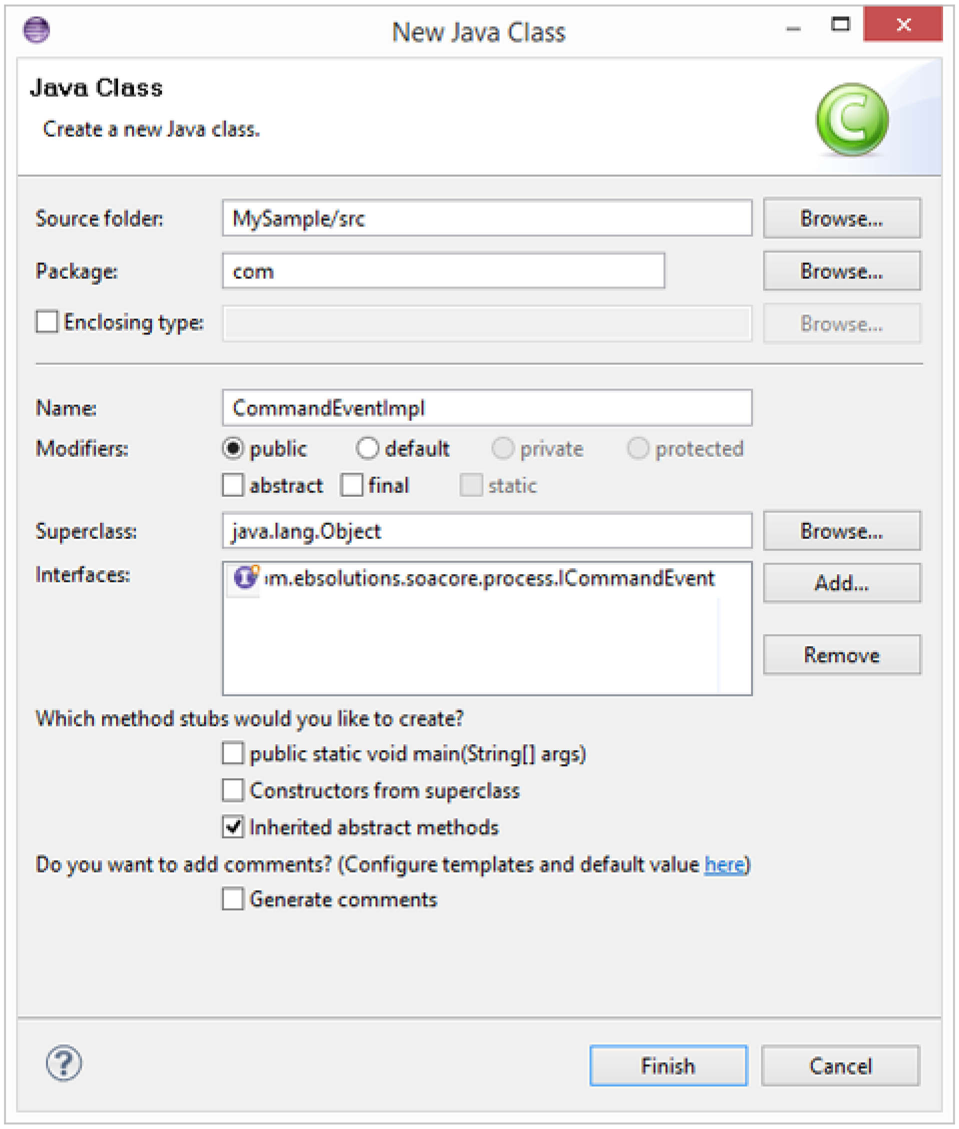
7. Enter *ICommandEvent* in the Choose interfaces: field, select the matching item, then pick **Add**. Pick **OK** when finished.



ICommandEvent - com.ebsolutions.soacore.process

com.ebsolutions.soacore.proc...iman\racless\Cat2soacore.jar

8. Pick **Finish**.



Source folder: MySample/src
Package: com
☐ Enclosing type:
Name: CommandEventImpl
Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces: ım.ebsolutions.soacore.process.ICommandEvent
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

Result:

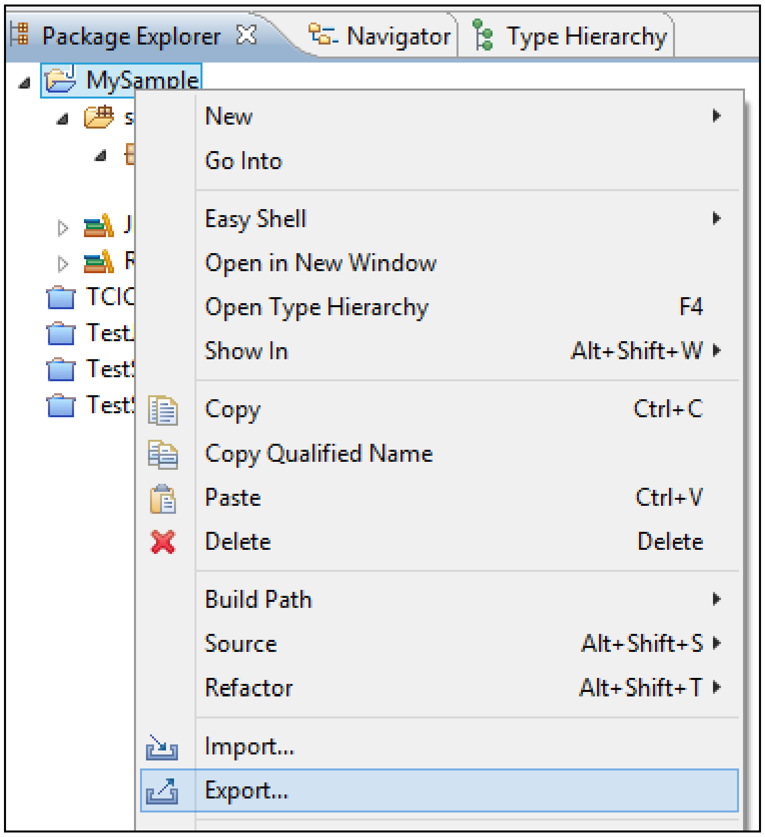
```
<code>
public class CommandEventImpl implements ICommandEvent {
     @Override
     public void onDone(OnDoneEventArgs arg0) {
           // TODO Auto-generated method stub
     }

     @Override
     public void onError(OnErrorEventArgs arg0) {
           // TODO Auto-generated method stub
     }

     @Override
     public void onStart(OnStartEventArgs arg0) {
           // TODO Auto-generated method stub
     }

} </code>
```

9. Modify the onDone and onStart functions:

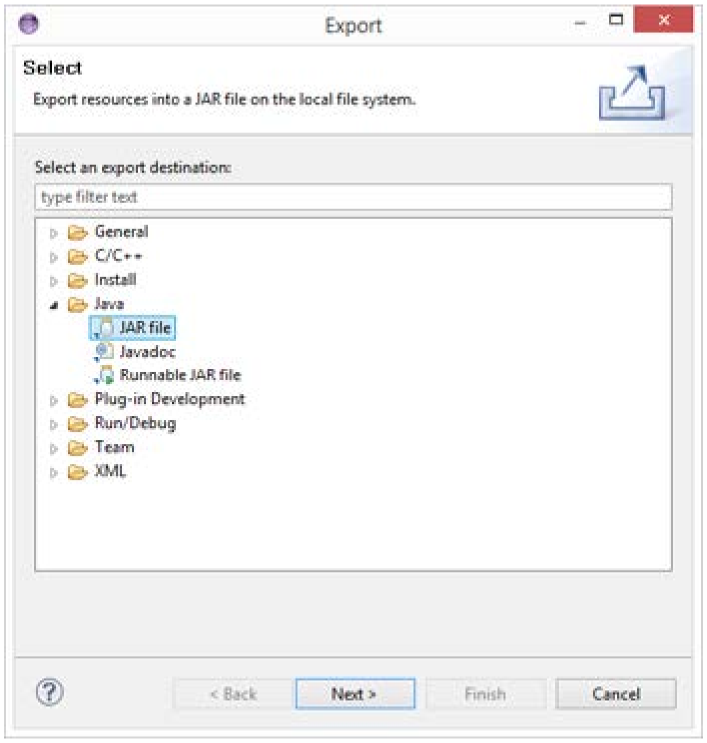
```
<code>
     @Override
     public void onDone(OnDoneEventArgs arg0) {
          System.out.println("CommandEventImpl::process is
finished");
     }

     @Override
     public void onStart(OnStartEventArgs arg0) {
           System.out.println("CommandEventImpl::process is
started");
     }
</code>
```

Generate a Library

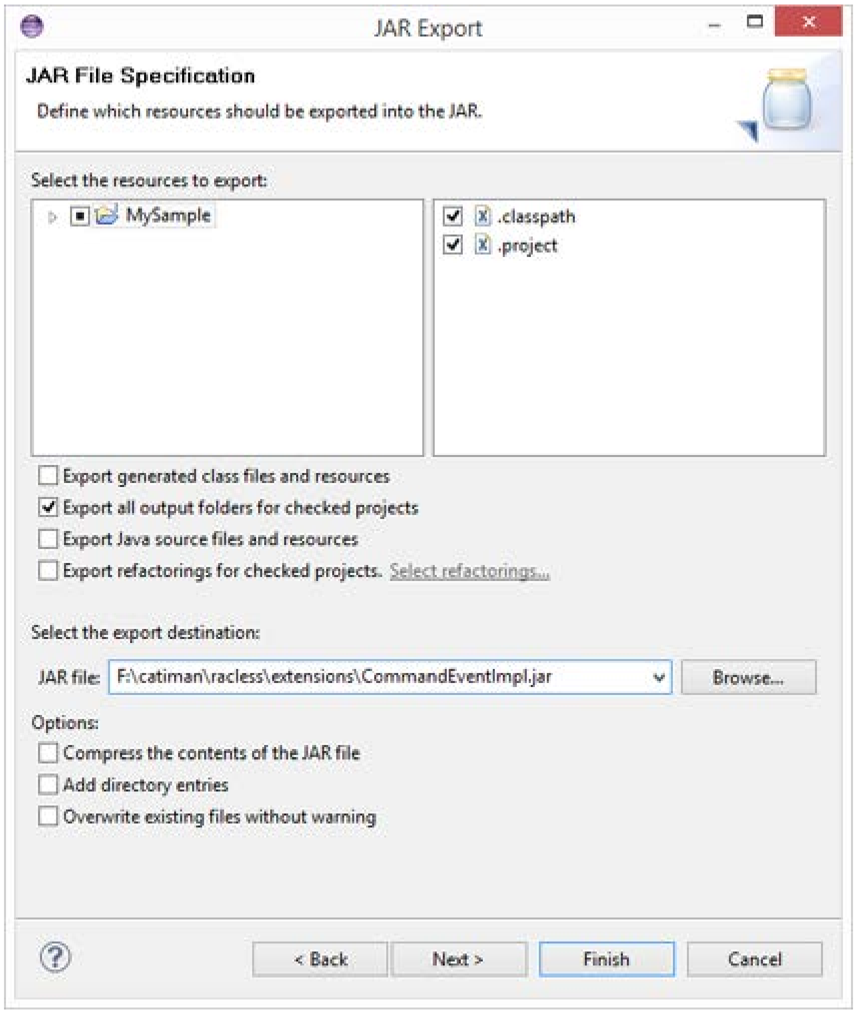
10. Highlight the project folder, then right-click and pick **Export**.



11. Toggle the **Java** folder open, select *JAR file*, then pick **Next**.



12. Select the project then enter a name and library location in the **JAR file** field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.


JAR Export
JAR File Specification
Define which resources should be exported into the JAR.
Select the resources to export:
MySample
.classpath
.project

- [ ] Export generated class files and resources
- [x] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

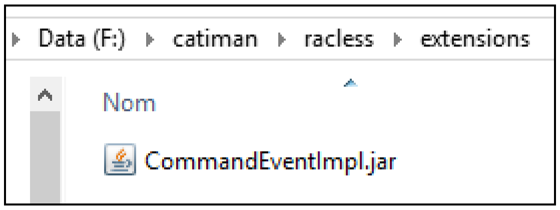
JAR file: F:\catiman\racless\extensions\CommandEventImpl.jar

Options:

- [ ] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

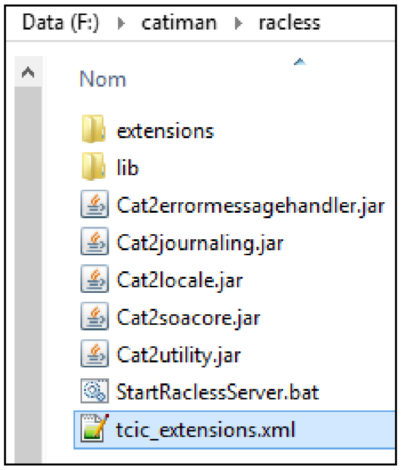
Result:

The user library is created and automatically loads during RACLess processes.



Indicate which interface has been implemented

13. Edit the *tcic_extnsions.xml* file within the *%TCICV5_DIR%\racless* directory to indicate which interface has been implemented.



- extensions
- lib
- Cat2errormessagehandler.jar
- Cat2journaling.jar
- Cat2locale.jar
- Cat2soacore.jar
- Cat2utility.jar
- StartRaclessServer.bat
- tcic_extensions.xml

Each implementation is referenced under a specific `<entry>` tag; `id` corresponds to the interface that has been implemented, `handler` corresponds to the implementation definition (its package and class) and `filename` corresponds the name of the user library.

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
      <entry
      id="com.ebsolutions.soacore.process.ICommandEvent"
      filename="CommandEventImpl.jar"
      handler="com.CommandEventImpl">
      </entry>
</extensions>
```

Several `<entry>` tags for the same interface may be defined, but the implementations must be different. Each implementation is called by the related process. For example,

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
     <entry
       id="com.ebsolutions.soacore.extensions.ICatiaUserExit"
       filename="CatiaUserExitImpl.jar"
       handler="com.CatiaUserExitImpl">
     </entry>
     <entry
       id="com.ebsolutions.soacore.process.ICommandEvent"
       filename="sample3.jar"
       handler="mysample.MyImpl3">
     </entry>
     <entry
       id="com.ebsolutions.soacore.process.ICommandEvent"
       filename="CommandEventImpl.jar"
       handler="com.CommandEventImpl">
     </entry>
</extensions>
```

Each implementation is called by the related process. Refer to the Import RACLess section for more details related to specific implementations.

In this sample, traces are written to the *tcic_racless_out_<timestamp>.log* file under the *%TEMP%* directory. Traces related to onStart calls are shown in red and traces related to onDone calls are shown in blue after files are imported.

```
[…]
Command sent [SaveFilesCommand]
Command dispatched [SaveFilesCommand]
callRACLessCommand com.ebsolutions.soacore.process.SaveFilesCommand
CommandEventImpl::process is started
TcICSOAServices.getTabStr(result_file)
TcICSOAServices.getTabStr(import_log_file)
CommandEventImpl::process is finished
send response [SEMAPH]
[…]
```

7.2.4.2 Examples of RACLess Customizations

Refer to the following table for a few examples of RACLess processes which can be customized. A complete list can be found here.

Process	Available Customization
Add CATShape	ICommandEvent: onStart, onError, onDone

Process	Available Customization
Check In/Check Out from CATIA	ICommandEvent: onStart, onError, onDone
Check- in/Check-out menu in Teamcenter	From the CATIAV5 Teamcenter menu: • ICommandEvent: onStart, onError, onDone From a sample and tcic_selection.xml: • ICommandEvent: onStart, onError, onDone • Inheritance: DoCheckAction.java • preProcessTasks • postProcessTasks • coreProcess

Process	Available Customization
RACLess Import	Use the UserImportCommand.java API provided • CommandEvent: onStart, onError, onDone From a sample: • Inheritance: RACLessProcessAbstract.java • preProcessTasks • postProcessTasks • coreProcess
RACLess Export	Use the given UserExportCommand.java API From a sample: • Inheritance: RACLessProcessAbstract.java • preProcessTasks • postProcessTasks • coreProcess

Process	Available Customization
RACLess Load, Update, Save	Use the given API UserLUSCommand.java ICommandEvent: onStart, onError, onDone From a sample: • Inheritance: RACLessProcessAbstract.java • preProcessTasks • postProcessTasks • coreProcess
Synchronize Active Shapes	ICommandEvent: onStart, onError, onDone
Other commands launched from CATIA	ICommandEvent: onStart, onError, onDone

7.2.4.3 Check in/Check Out Customization

RACLess ProcessAbstract Implementation
Use a Java Project to Compile and Install the Sample

7.2.4.3.1 RACLessProcessAbstract Implementation

This class inherits from RACLessProcessAbstract class:

- preProcessTasks() method is empty, by default
- postProcessTasks() method displays a message
- initProcessConfiguration() method defines configuration through the UserExportCommand.getDefaultProcessConfiguration() API
- coreProcess() method:
    - reads the *tcic_selection.xml* file

- launches DoCheckInAction or DoCheckOutAction on the BOM Lines (calls DoCheckAction command with check mode = CI if mode is set to "CI", calls DoCheckAction command with check mode = CO if mode is set to "CO")

Code sample

```
package com;
import javax.swing.JOptionPane;
import com.ebsolutions.soacore.process.DoCheckAction;
import com.ebsolutions.soacore.process.ProcessConfiguration;
import com.ebsolutions.soacore.process.RACLessProcessAbstract;

public class CustoCICO extends RACLessProcessAbstract {
     @Override
      protected int coreProcess() {
            try
            {
                  String CI_ACTION = "mode=CI";
                  String CO_ACTION = "mode=CO";
                  DoCheckAction action = new
DoCheckAction();

//execute a Check-In Action (DoCheckInAction command is launched),
if "mode=CI"

//execute a Check-Out Action (DoCheckInAction command is launched),
if "mode=CO"

 action.executeProcess(new String[]{CI_ACTION});
            } catch (Exception e) {
                  e.printStackTrace();
            }
           return 0;
      }
      @Override
      protected int initProcessConfiguration() {
            //Get the default configuration for a silent process
           _process_configuration =
ProcessConfiguration.getDefaultSilentProcessConfiguration();
            _process_configuration.initLogFile("RACLessProcessSampl
e.log", true);
           return 0;
      }
      @Override
      protected int postProcessTasks() {
            JOptionPane.showMessageDialog(null, "CustoCICO is
complete");
            return 0;
      }
      @Override
      protected int preProcessTasks() {
            // TODO Auto-generated method stub
            return 0;
      }
}
```

Compilation and Installation of the Sample

Perform the following steps to compile and install the sample.

Refer to the notes below before proceeding.

1. Update the following variables in the batch file (compile.bat), if necessary:
   a. TC_ROOT
   b. FMS_HOME
   c. JAVA_HOME
   d. TCICV5_DIR
2. Modify the name of the class, if necessary.
3. Modify the name of the jar, if necessary.
4. The compiled .jar file is now available in the current directory.

Install the .jar file

The .jar file can be installed under *%TCICV5_DIR%\racless\extensions* directory. In this case, the jar is automatically added to the TcIC classpath.

The .jar file can also be installed under another directory. In this case, add the new directory to the TcIC class path.

5. For example, the *Cat2SampleSilentExport.jar* file is compiled under the *F:\Temp* directory. Modify the *%TCICV5_DIR%\racless\StartRaclessServer.bat* file as follows:
   a. Add the .jar file to the classpath RACLESS_CLASSPATH
      set
      RACLESS_JAVA_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fmsclientcache.jar
      set
      RACLESS_JAVA_CLASSPATH=%RACLESS_JAVA_CLASSPATH%;%FMS_HOME%\jar\fscjavaclientproxy.jar
      set
      RACLESS_JAVA_CLASSPATH=%RACLESS_JAVA_CLASSPATH%;%FMS_HOME%

\jar\fccjavaclientproxy.jar
set
RACLESS_JAVA_CLASSPATH=%RACLESS_JAVA_CLASSPATH%;%FMS_HOME%
\jar\fmsutil.jar

Execution

6. Update the following variables in the ***runCICOCommand.bat*** file.

   a. USER

   b. PASSWORD

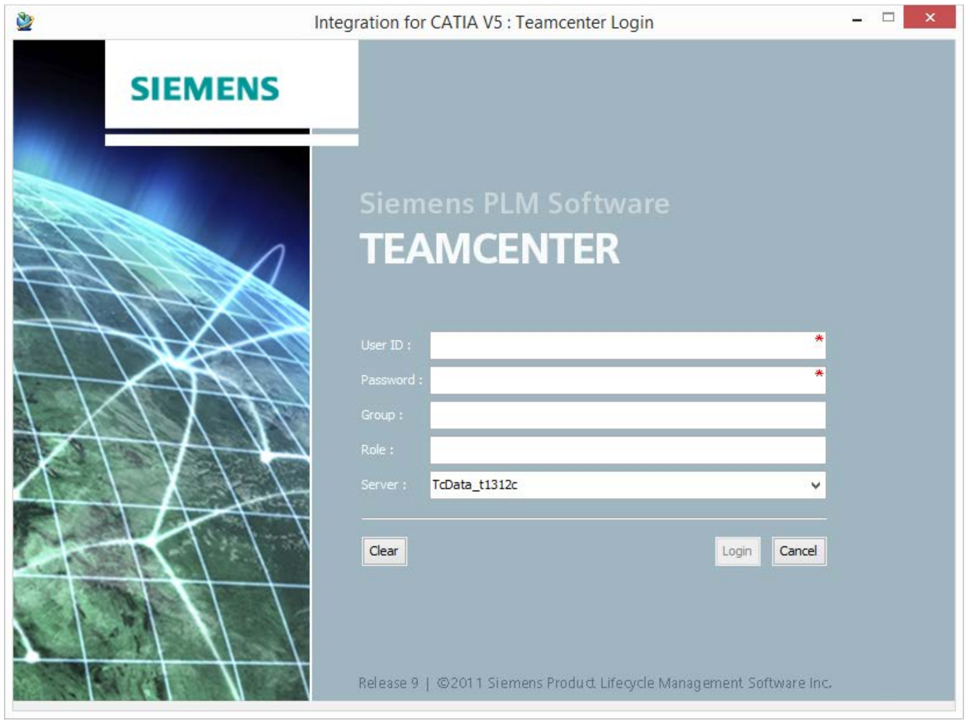
   c. HOST

   d. TCICV5_DIR

Notes

When the USER, PASSWORD and HOST variables are set with an empty value, the login RACLess panel displays. These variables should not contain the **"** character. For example,

```
set USER=pbu
set PASSWORD=
set HOST= iiop:localhost:1572/TcServer1
```


Integration for CATIA V5 : Teamcenter Login
SIEMENS
Siemens PLM Software
TEAMCENTER
User ID :
Password :
Group :
Role :
Server :
TcData_t1312c
Clear
Login
Cancel
Release 9 | ©2011 Siemens Product Lifecycle Management Software Inc.

If the class does not correspond to the **com.CustoCICO** class, then modify the following line with the correct class name:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.CustoCICO 0 11400

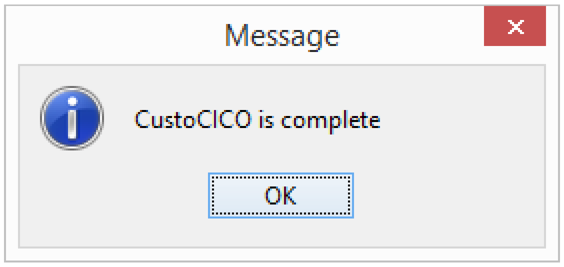
When the socket port has been changed, modify the following line:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.CustoCICO 0 11400

And modify the *%TCICV5_DIR%\env\tcic_var_env.xml* file:

```
<TcIC_socket_port>
11400
</TcIC_socket_port>
```

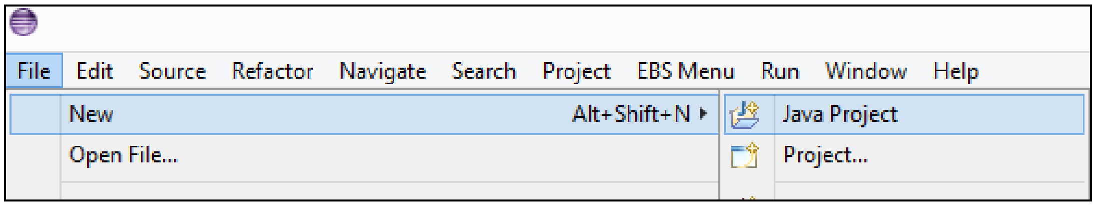
The *tcic_selection.xml* file can now be created and *runCustcoCICO.bat* can be launched.


Message

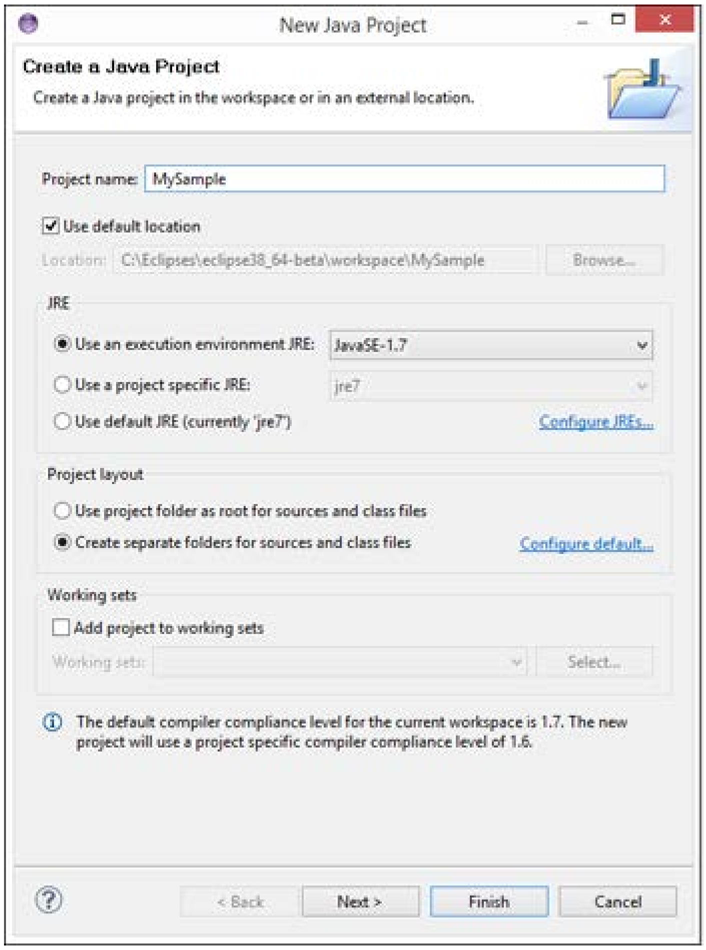
CustoCICO is complete

7.2.4.3.2 *Use a Java Project to Compile and Install the Sample*

1. In Eclipse, create a new Java project.



2. Enter the **Project name** then select the compatible JRE version: **Java 1.6** with Teamcenter 9.1.2.9 or **Java 1.7** for Teamcenter 10.1.2. Pick **Next** to continue.



Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: MySample

[✓] Use default location

Location: C:\Eclipses\eclipse38_64-beta\workspace\MySample

JRE

(●) Use an execution environment JRE: JavaSE-1.7

( ) Use a project specific JRE: jre7

( ) Use default JRE (currently 'jre7')

Project layout

( ) Use project folder as root for sources and class files

(●) Create separate folders for sources and class files

Working sets

[ ] Add project to working sets

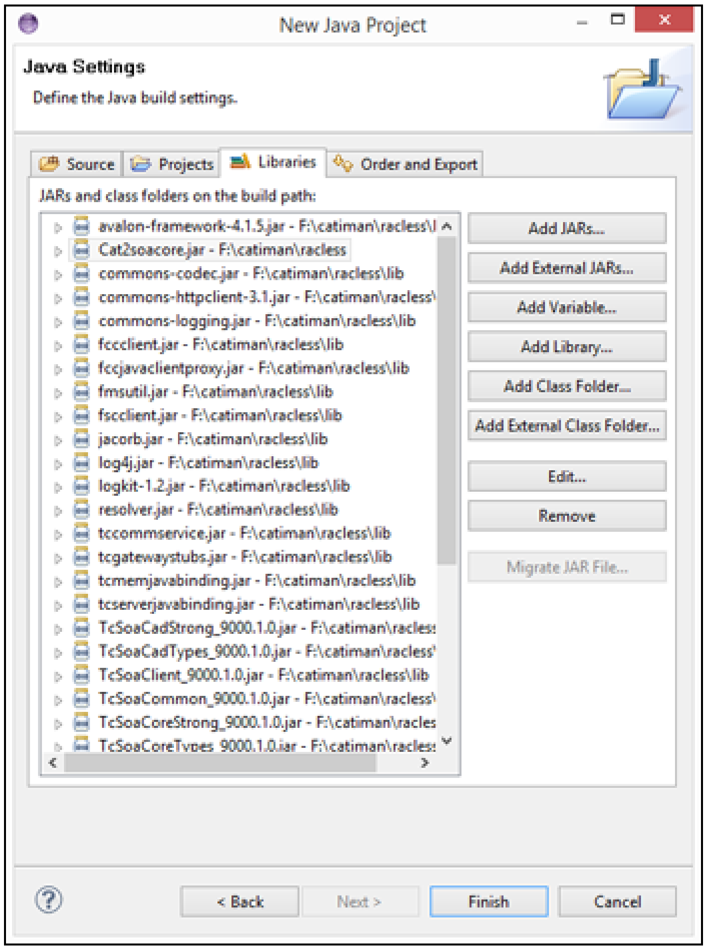
Working sets:

The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.6.

3. Select the **Libraries** tab, then select the libraries below required for the project. Pick **Finish** when done.

   a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

   b. Cat2soacore.jar library contained in *%TCICV5_DIR%\racless.*



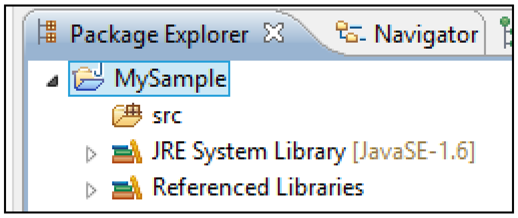
Java Settings

Define the Java build settings.

JARs and class folders on the build path:

```
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless
```

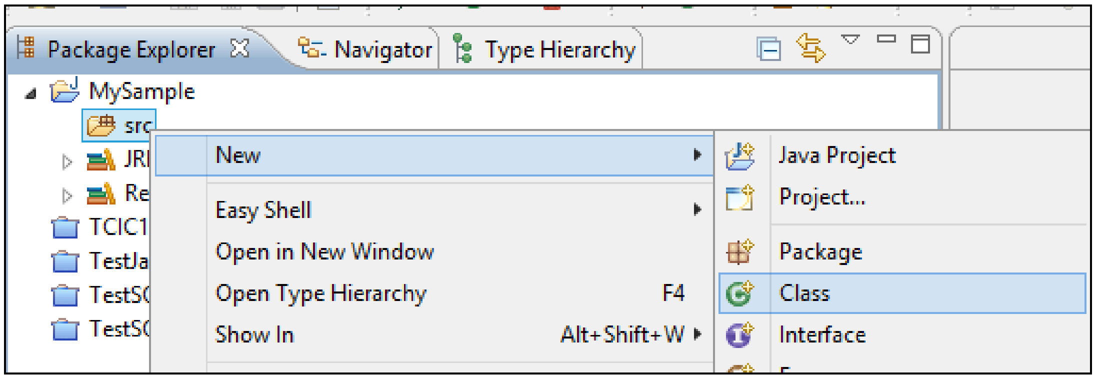
Result:



Overload the executeProcess function

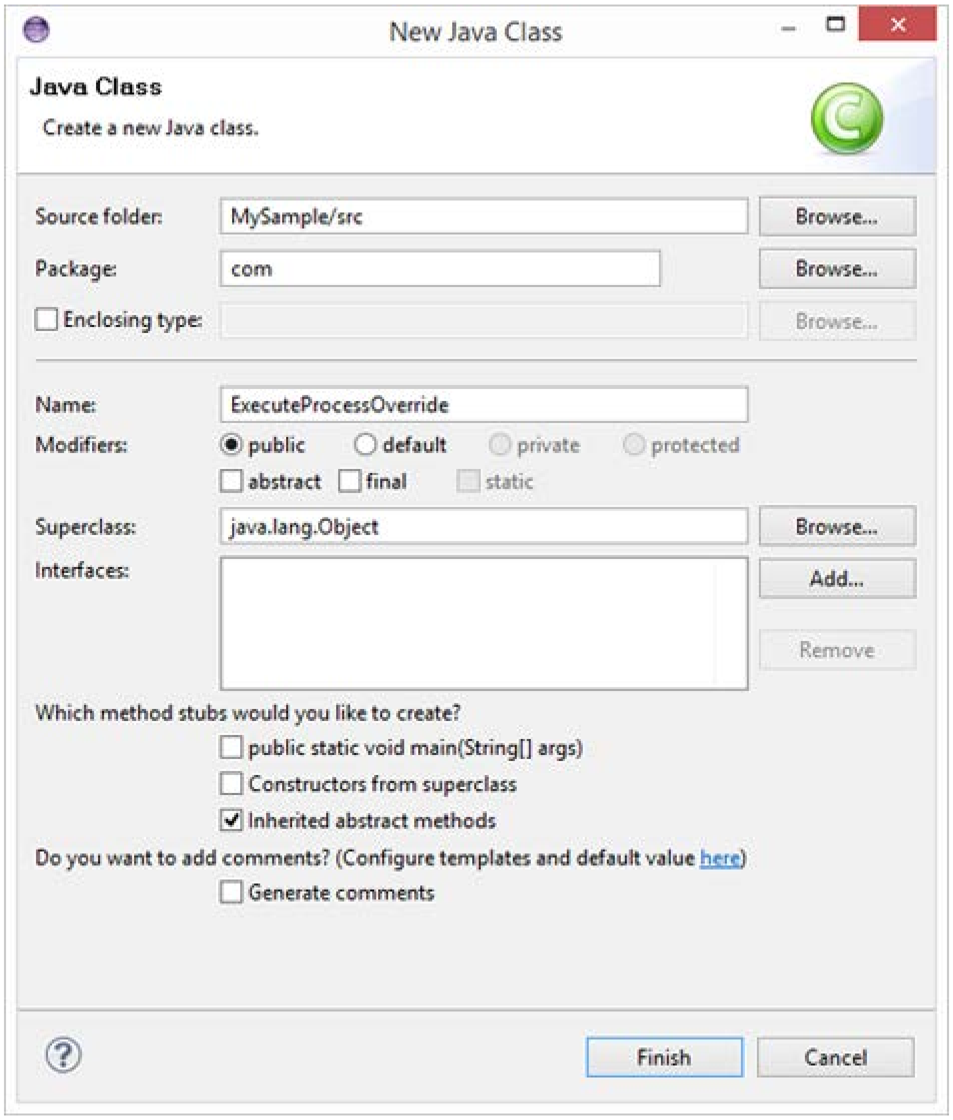
A customized process must always be placed into a specific package such as com.MyCusto

7. Select the **src** folder, right-click and then select **New >> Class**.



8. Complete the **Package:** and **Name:** fields on the New Java Class window.

9. Untoggle **public static void main(String[] args)** and toggle the **Inherited abstract methods** option on. Pick the **Browse...** button next to the Superclass: field.



Source folder: MySample/src

Package: com

☐ Enclosing type:

Name: ExecuteProcessOverride

Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static

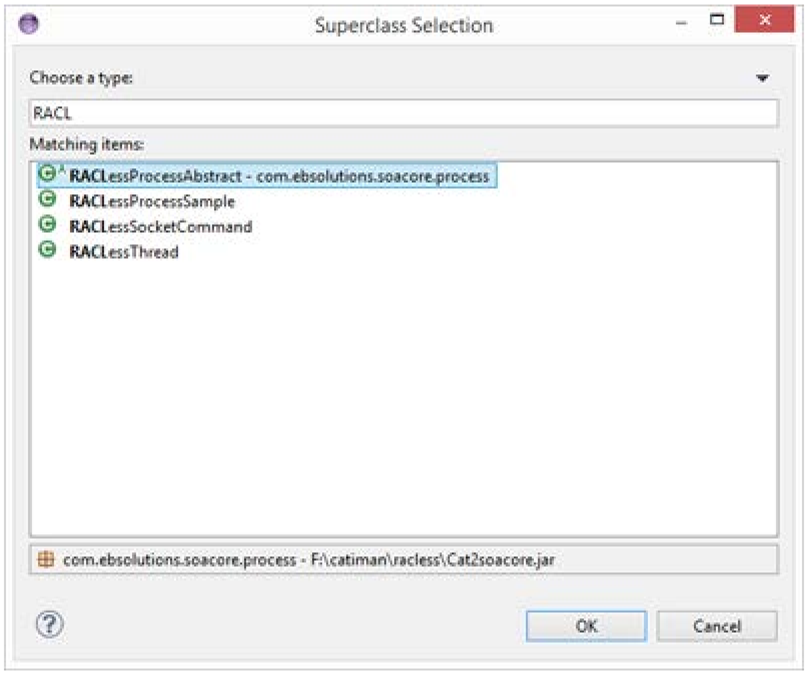
Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

10. Enter *RACLessProcessAbstract* in the Choose a type: field. Select **RACLessProcessAbstract** from the resulting Matching items box, then pick **OK**.



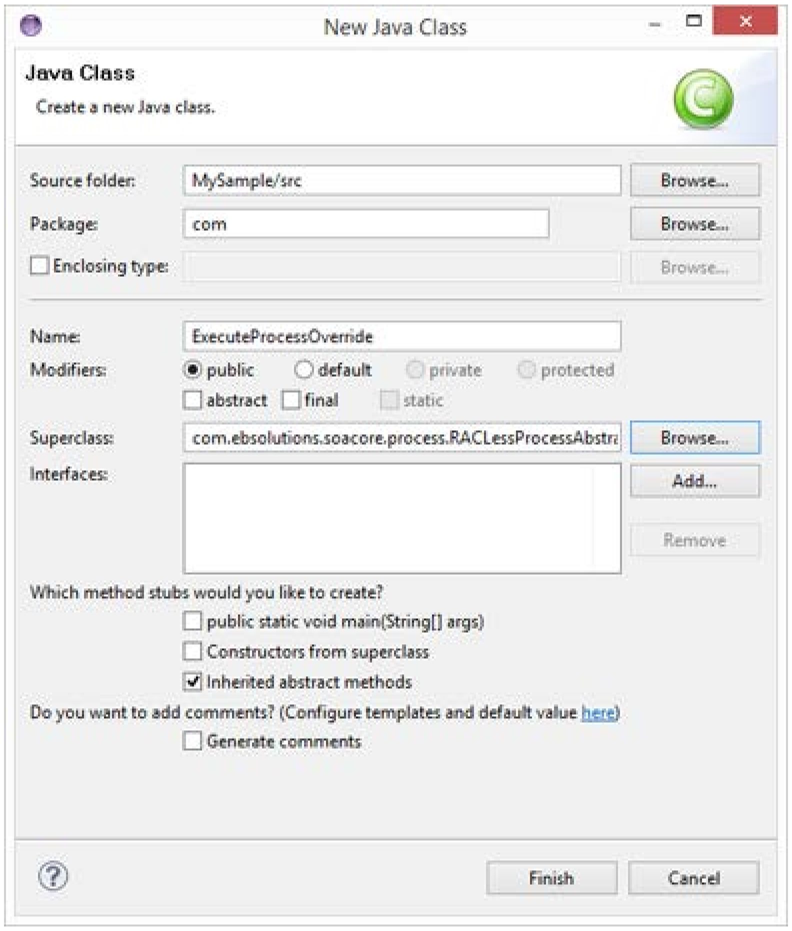
RACL

Matching items:

RACLessProcessAbstract - com.ebsolutions.soacore.process
RACLessProcessSample
RACLessSocketCommand
RACLessThread

com.ebsolutions.soacore.process - F:\catiman\racless\Cat2soacore.jar

11. Pick **Finish**.



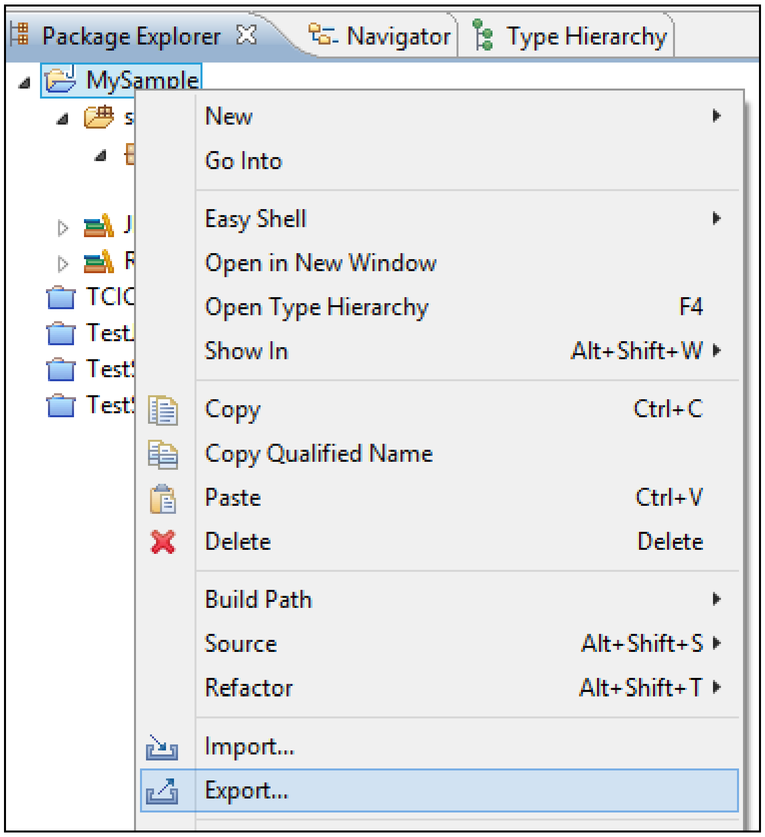
Source folder: MySample/src
Package: com
☐ Enclosing type:
Name: ExecuteProcessOverride
Modifiers: ● public ○ default ○ private ○ protected
☐ abstract ☐ final ☐ static
Superclass: com.ebsolutions.soacore.process.RACLessProcessAbstr
Interfaces:
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

Implement functions as follows:

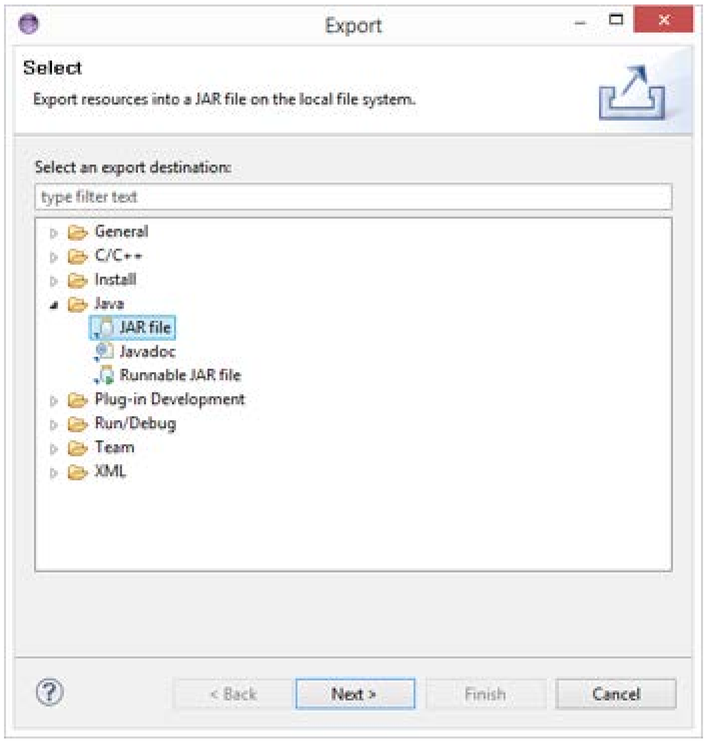
```
public class CustoCICO extends RACLessProcessAbstract {
     @Override
     protected int coreProcess() {
           try
           {
                 String CI_ACTION = "mode=CI";
                 String CO_ACTION = "mode=CO";
                 DoCheckAction action = new DoCheckAction();
                 //execute a Check-In Action ("mode=CI") regarding
the tcic_selection.xml content
                 //for a Check-Out Action, set the String value to
"mode=CO"
                 action.executeProcess(new String[]{CI_ACTION});
           } catch (Exception e) {
                 e.printStackTrace();
           }
           return 0;
     }
     @Override
     protected int initProcessConfiguration() {
           //Get the default configuration for a silent process
           _process_configuration =
ProcessConfiguration.getDefaultSilentProcessConfiguration()
    _process_configuration.initLogFile("RACLessProcessSample.log",
true);
           return 0;
     }
     @Override
     protected int postProcessTasks() {
           JOptionPane.showMessageDialog(null, "CustoCICO is
complete");
           return 0;
     }
     @Override
     protected int preProcessTasks() {
           // TODO Auto-generated method stub
           return 0;
     }
}
```

Generate the Library

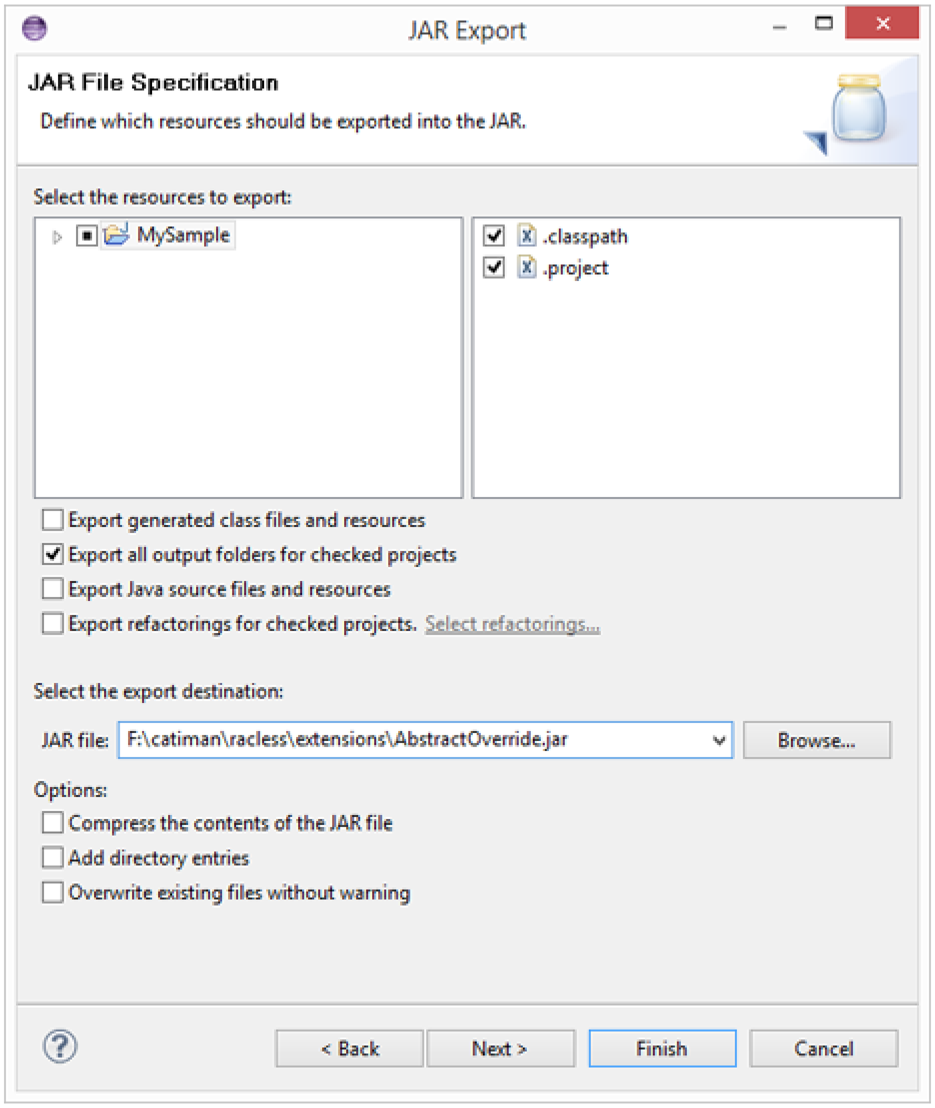
12. Highlight the project folder, then right-click and pick **Export**.



13. Toggle the **Java** folder open, select *JAR file*, then pick **Next**.



14. Select the project then enter a name and library location in the **JAR file** field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [■] MySample
- [✓] .classpath
- [✓] .project

- [ ] Export generated class files and resources
- [✓] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

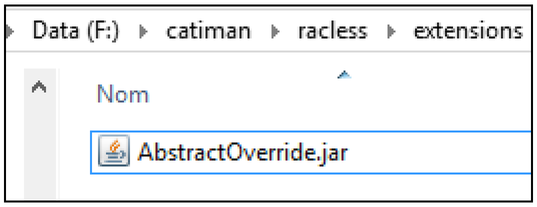
Select the export destination:

JAR file: F:\catiman\racless\extensions\AbstractOverride.jar

Options:

- [ ] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

The library automatically loads when the RACLess process launches.


Data (F:) ▸ catiman ▸ racless ▸ extensions

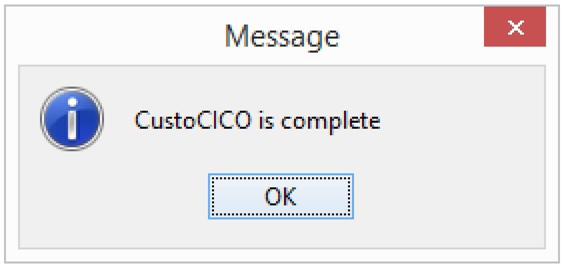
Nom

AbstractOverride.jar

Test the Implementation

1. Use the *%TCICV5_DIR%\racless\StartRaclessServer.bat* command.

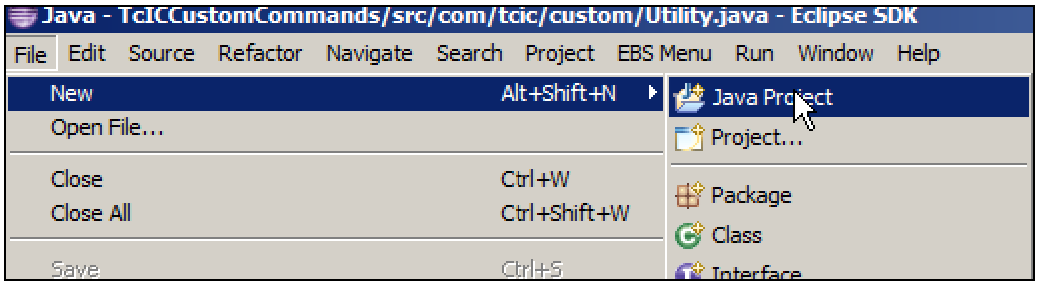
2. Launch following instruction in a command line: *%TCICV5_DIR%\bin\WIN32\SocketV5.exe com.CustoCICO 0 11400.* The following message displays:



7.2.4.4 Save Manager and State Details Customization

Use these steps to implement pre/post actions on buttons in the Save Manager and State Details panel.

1. Pick **File >> Java Project** to create a new Java project.

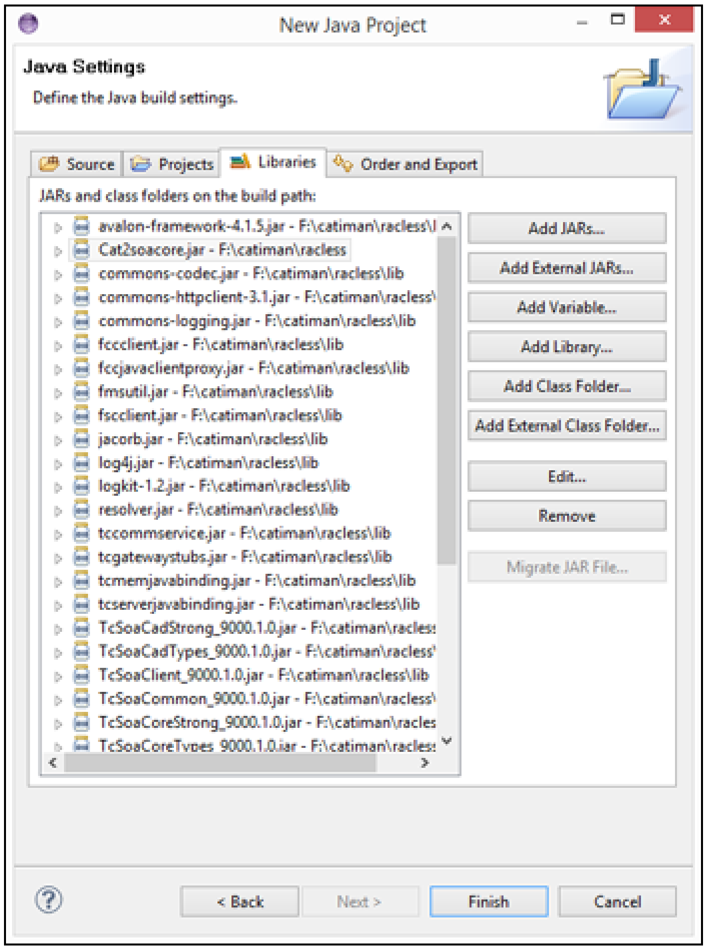


2. Enter the project name and verify the correct Java version In the Java Project dialog. Pick **Next** to continue.

3. Select the **Libraries** tab, then select the libraries below required for the project. Pick **Finish** when done.

   a. Teamcenter SOA libraries contained in *%TCICV5_DIR%\racless\lib*

   b. Cat2soacore.jar and Cat2uimanager.jar library contained in *%TCICV5_DIR%\racless.*



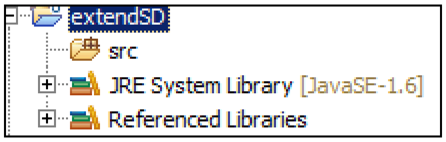
Java Settings

Define the Java build settings.

JARs and class folders on the build path:

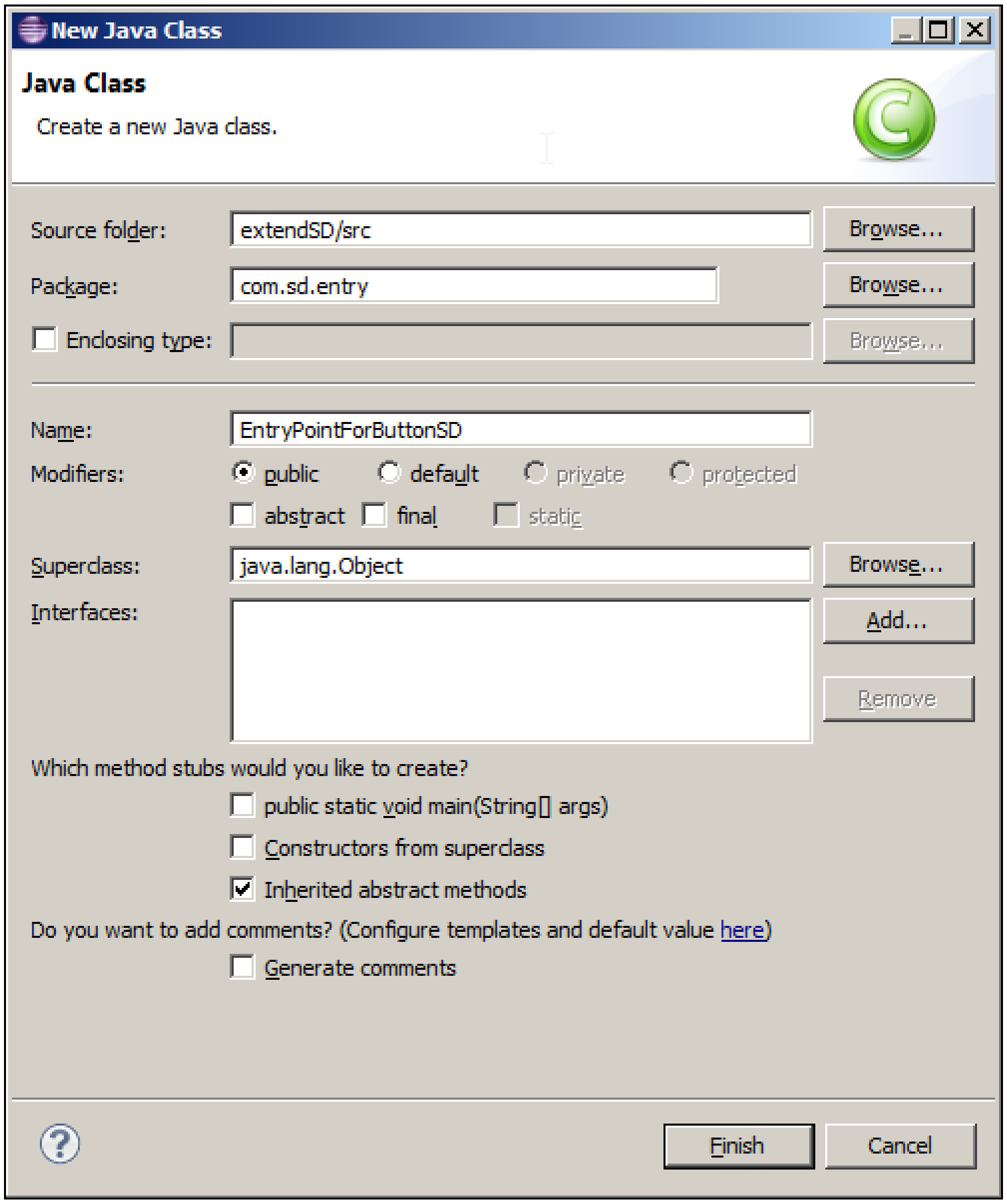
```
avalon-framework-4.1.5.jar - F:\catiman\racless\l
Cat2soacore.jar - F:\catiman\racless
commons-codec.jar - F:\catiman\racless\lib
commons-httpclient-3.1.jar - F:\catiman\racless\
commons-logging.jar - F:\catiman\racless\lib
fccclient.jar - F:\catiman\racless\lib
fccjavaclientproxy.jar - F:\catiman\racless\lib
fmsutil.jar - F:\catiman\racless\lib
fscclient.jar - F:\catiman\racless\lib
jacorb.jar - F:\catiman\racless\lib
log4j.jar - F:\catiman\racless\lib
logkit-1.2.jar - F:\catiman\racless\lib
resolver.jar - F:\catiman\racless\lib
tccommservice.jar - F:\catiman\racless\lib
tcgatewaystubs.jar - F:\catiman\racless\lib
tcmemjavabinding.jar - F:\catiman\racless\lib
tcserverjavabinding.jar - F:\catiman\racless\lib
TcSoaCadStrong_9000.1.0.jar - F:\catiman\racless
TcSoaCadTypes_9000.1.0.jar - F:\catiman\racless\
TcSoaClient_9000.1.0.jar - F:\catiman\racless\lib
TcSoaCommon_9000.1.0.jar - F:\catiman\racless\
TcSoaCoreStrong_9000.1.0.jar - F:\catiman\racles
TcSoaCoreTypes_9000.1.0.jar - F:\catiman\racless
```

Result in Eclipse:



```
extendSD
    src
    JRE System Library [JavaSE-1.6]
    Referenced Libraries
```

4. In Eclipse, create a class in the **com.sd.entry** package, then pick **Finish**. For example, *EntryPointForButtonSD*.



Source folder: extendSD/src
Package: com.sd.entry
☐ Enclosing type:
Name: EntryPointForButtonSD
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

5. In the newly created class, implement the *com.ebsolutions.savemanager.extension.IEntryPointForButtonAction* interface and abstract methods. Eclipse will prompt when adding empty implementations for abstract methods.

```
package com.sd.entry;
import com.ebsolutions.savemanager.SaveManagerApplication;
import
com.ebsolutions.savemanager.extension.IEntryPointForButtonAction;
import com.ebsolutions.statedetails.StateDetailsApplication;
public class EntryPointForButtonSD  implements
```

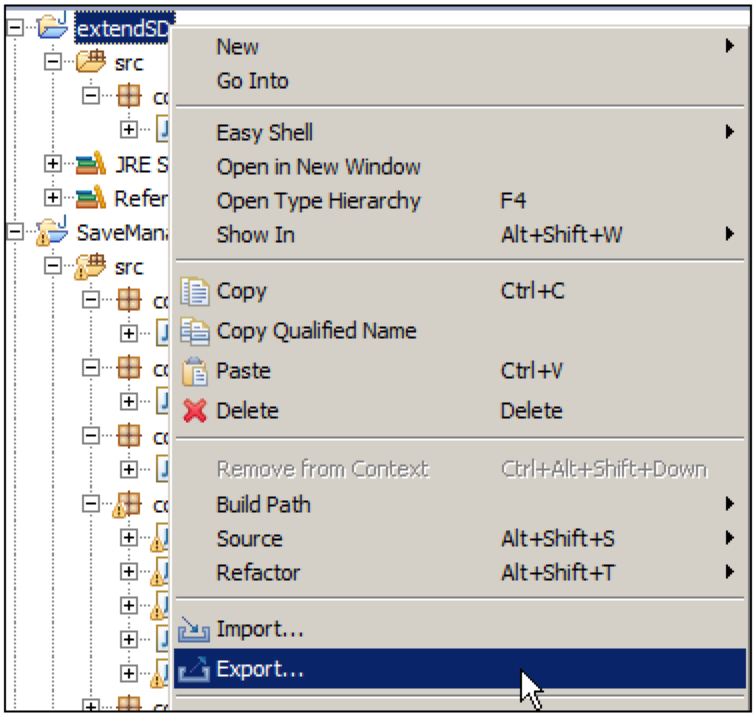
```
IEntryPointForButtonAction{
   @Override
   public boolean userEntryPreButtonAction(String paramString,
           SaveManagerApplication paramSaveManagerApplication) {
       //  TODO Auto-generated method stub
       return false;
   }
   @Override
   public boolean userEntryPostButtonAction(String paramString,
           SaveManagerApplication paramSaveManagerApplication) {
       // TODO Auto-generated method stub
       return false;
   }
   @Override
   public boolean userEntryPreButtonAction(String paramString,
           StateDetailsApplication paramStateDetailsApplication) {
       // TODO Auto-generated method stub
       return false;
   }
   @Override
   public boolean userEntryPostButtonAction(String paramString,
           StateDetailsApplication paramStateDetailsApplication) {
       // TODO Auto-generated method stub
       return false;
   }
}
```

6. Modify the default empty implementation to log the clicked button ID for the Save Manager and State Details buttons.

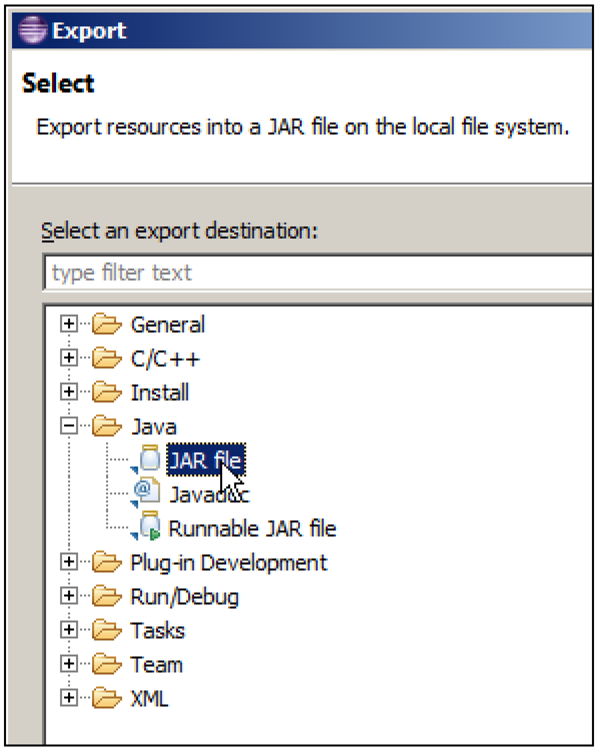
```
package com.sd.entry;
import com.ebsolutions.savemanager.SaveManagerApplication;
import
com.ebsolutions.savemanager.extension.IEntryPointForButtonAction;
import com.ebsolutions.statedetails.StateDetailsApplication;
public class EntryPointForButtonSD  implements
IEntryPointForButtonAction{
    /**
     * @param btnID The clicked button in Savemanager application
     * @param paramSaveManagerApplication SavemanagerApplication,
     * can be used to get any information from the application
     */
    @Override
    public boolean userEntryPreButtonAction(String btnID,
            SaveManagerApplication paramSaveManagerApplication) {
        System.out.println("EntryPointForButtonSD.userEntryPreButto
nAction() For SaveManager");
        // here is possible button ids
        if(btnID == "create_jt_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
```

```
        } else if(btnID == "compute_geo_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        }  else if(btnID == "compute_geo_for_main_body_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        } else if(btnID == "save_active_shapes_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        } else if(btnID == "hide_show_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        } else if(btnID == "check_in_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        } else if(btnID == "force_save_flag") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        }else if(btnID == "browse_destination_folder_id") {
            System.out.println("SaveManager: Start "+btnID+"
process");
        } else {
            System.out.println("START "+btnID +" process");
        }
        // return True if you want to continue executing the
action, False if not
        return true;
    }
    @Override
    public boolean userEntryPostButtonAction(String btnID,
            SaveManagerApplication paramSaveManagerApplication) {
        System.out.println("EntryPointForButtonSD.userEntryPostButt
onAction() for SaveManager");
        System.out.println("End "+btnID +" process");
        return true;
    }
    @Override
    public boolean userEntryPreButtonAction(String btnID,
            StateDetailsApplication paramStateDetailsApplication) {
        System.out.println("EntryPointForButtonSD.userEntryPreButto
nAction() For StateDetailsApplication");
        if (btnID.equalsIgnoreCase("check_in")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("check_out")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("highlight_selected")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("highlight_parent")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("refresh")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("replace_by_revision")) {
            System.out.println("START "+btnID +" process");
        } else if (btnID.equalsIgnoreCase("update")) {
            System.out.println("START "+btnID +" process");
        } else {
            System.out.println("START "+btnID +" process");
        }
        return true;
    }
    @Override
    public boolean userEntryPostButtonAction(String btnID,
            StateDetailsApplication paramStateDetailsApplication) {
        System.out.println("EntryPointForButtonSD.userEntryPostButt
onAction() For StateDetailsApplication");

        System.out.println("END "+btnID+" process");

        return true;
    }
```

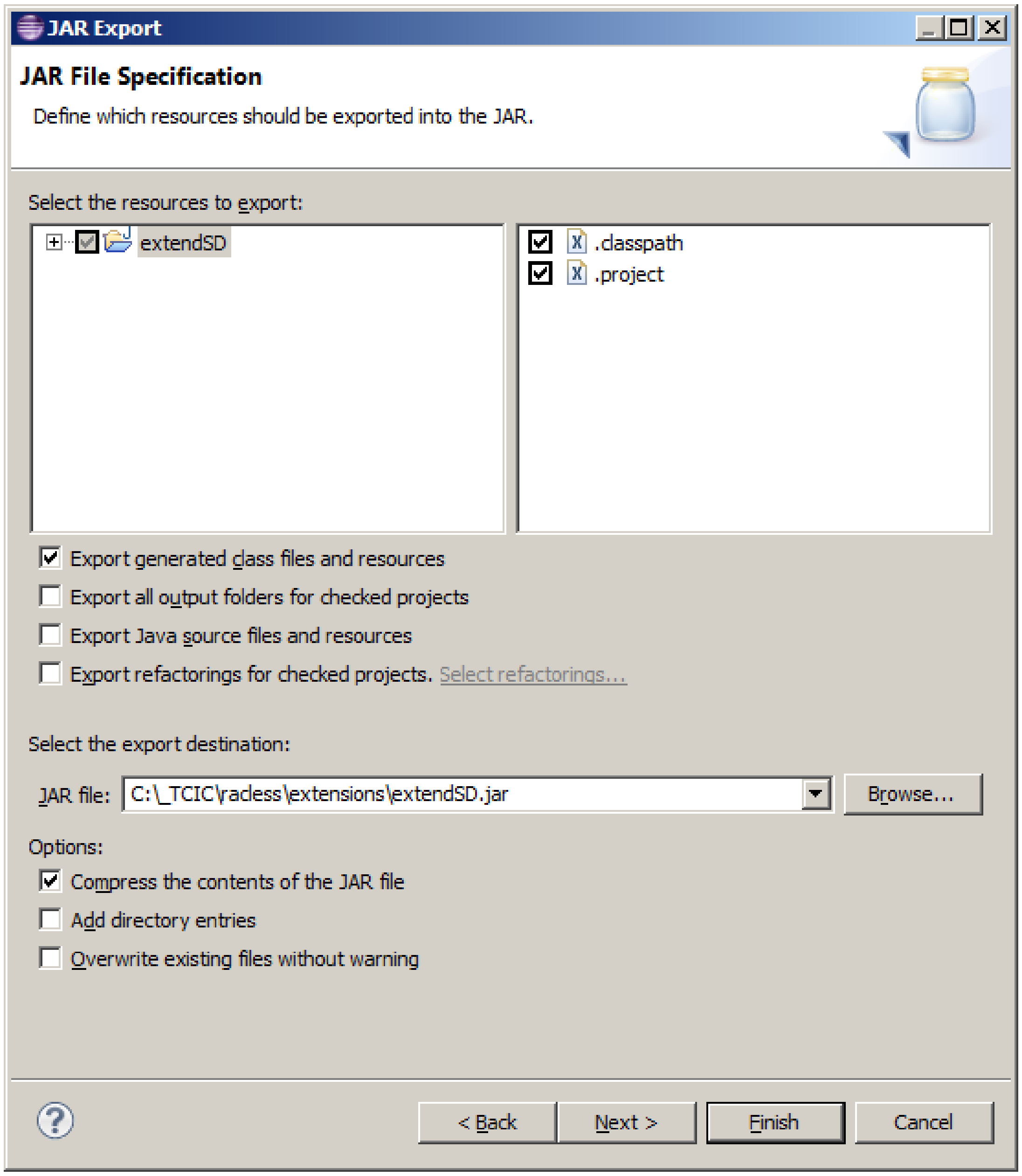
7. After all modifications are complete, highlight the project folder, right-mouse click then pick **Export**.



8. Toggle the Java folder open, select the **JAR** file, then pick **Next**.



9. Select the project, then enter a name and library location in the JAR file field. Place the library under the *%TCICV5_DIR%\racless\extensions* directory.


JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [✓] extendSD
- [✓] .classpath
- [✓] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: C:_TCIC\racless\extensions\extendSD.jar

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

10. Modify the *tcic_extension.xml* file and the information about the implemented entry point. For each entry tag in this file, ensure the following properties are included:

    a. `id` : the interface/class defining the entry point, it is always a TcIC class

    b. `filename`: the JAR file containing the implementation of the entry point

    c. `handler`: the class that extends/implements the TcIC entry point

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
  <entry

id="com.ebsolutions.savemanager.extension.IEntryPointForButtonActio
n"
      filename="extendSD.jar"
      handler="com.sd.entry.EntryPointForButtonSD">
  </entry>
</extensions>
```

Test the Extension

Perform the following steps to test the **Save Manager** buttons.

1. In CATIA, create a CATPart and **Save**.
2. In the Save Manager panel:
    a. Select the line corresponding to the CATPart then pick **Assign**.
    b. Pick **Create JT**.
    c. Pick **Reset IDs**.
    d. Pick **Assign**.
    e. **Apply** and **Finish** the save.
3. View the *tcic_racless_out_*.log* log file to ensure the implementation is called for each clicked button.

```
........
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
START assign process
.......
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
.......
EntryPointForButtonSD.userEntryPostButtonAction() for SaveManager
End assign process
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
START create_jt_flag process
EntryPointForButtonSD.userEntryPostButtonAction() for SaveManager
End create_jt_flag process
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
START reset process
.......
EntryPointForButtonSD.userEntryPostButtonAction() for SaveManager
```

```
End reset process
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
START assign process
.......
EntryPointForButtonSD.userEntryPostButtonAction() for SaveManager
End assign process
.......
EntryPointForButtonSD.userEntryPreButtonAction() For SaveManager
START application_okAction process
EntryPointForButtonSD.userEntryPostButtonAction() for SaveManager
End application_okAction process
.......
```

Perform the following tests to test the State Details Panel button.

1. In CATIA, launch the Detailed States process on the CATPart created earlier for the Save Manager button test.

2. In the State Detail Panel, pick **Update**, then pick **Check-in**.

3. View the *tcic_racless_out_*.log* log file to ensure the implementation is called for each clicked button.

```
........
EntryPointForButtonSD.userEntryPreButtonAction() For
StateDetailsApplication
START update process
........
EntryPointForButtonSD.userEntryPostButtonAction() For
StateDetailsApplication
END update process
........
EntryPointForButtonSD.userEntryPreButtonAction() For
StateDetailsApplication
START check_in process
........
EntryPointForButtonSD.userEntryPostButtonAction() For
StateDetailsApplication
END check_in process
```

7.2.4.5 Silent Import

UserImportCommand API

SampleSilentImport Sample

The Import RACLess process is based on the Data.txt file to launch Import Spreadsheet creation and Import.

The Import RACLess pocess can also be based on parameters given through a command line, then connection information and documents to be imported can be set through parameters.

There are two different methods to launch import RACLess.

1. SampleImportCommandLineWin.bat: Does not permit customized pre/post actions and processes cannot be locked while the Create Import Spreadsheet and Import processes run.

2. runSilentImportCommand.bat. Permits customized pre/post actions and processes can be locked while the Create Import Spreadsheet and Import processes run.

Both ICommandEvent and ICatiaUserExit interfaces can be used in either methods.

	Available using the batch method	**Available using the RACLess method**
ICommandEvent	yes	yes
ICatiaUserExit	yes	yes
preProcessTasks	no	yes
postProcessTasks	no	yes
Lock/Unlock process	partial	yes

Method 1: Launch two CATScript processes

Refer to the *%TCICV5_DIR%\custom_client\SilentImport\SampleImportCommandLineWin.bat* file to proceed.

Ensure that this method is not so robust in the lock / unlock process management to cause the process to be locked between the spreadsheet creation and import, as it is when creating your own process.

This method does not permit implementing pre/post actions, such as the preProcessTasks and postProcessTasks methods of the RACLessProcessAbstract class.

SampleRACLessImport
SampleImportCommandLineWin.bat

SampleImportCommandLineWin.bat launches following steps:

- StartRACLessServer.bat file in a silent or non-silent mode to launch the RACLess connection. By default, the .bat file is launched in silent mode.
- CreateAllSpreadSheets.CATScript macro in batch mode.
- ImportAllAssemblies.CATScript in batch mode.
- CloseRACLessServer to close the RACLess connection.

Modify the following lines in the .bat file to customize the sample file.

```
Start "RACLess Server" /min cmd /c
"%TCICV5_DIR%\racless\StartRaclessServer.bat –silent %*"

"C:\Program Files\Dassault
Systemes\B19\intel_a\code\bin\catstart.exe" -run CNEXT -env
CATIA_V5R19_CAA -direnv "C:\Program Files\Dassault Systemes\CATEnv"
-object "-batch %WORK_DIR%\Import\%1  -macro
%TCICV5_DIR%\CATScript\CreateAllSpreadSheets.CATScript"

"C:\Program Files\Dassault
Systemes\B19\intel_a\code\bin\catstart.exe" -run CNEXT -env
CATIA_V5R19_CAA -direnv "C:\Program Files\Dassault Systemes\CATEnv"
-object "-batch %WORK_DIR%\Import\%1  -macro
%TCICV5_DIR%\CATScript\ImportAllAssemblies.CATScript"

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" CloseRaclessServer 0 11400
```

- Run SampleImportCommandLineWin.bat without any parameters so that it's based on the Data.txt file and the top level document defined in the Data.txt file is imported.

- Do not use the Data.txt file to customize the Import process. Instead, run SampleImportCommandLineWin.bat with the following **optional** arguments.
  - `-u=<user-id> -p=<password> -g=<group> -r=<role> -h=<host> -nl=<language> -proj=<project_id> -client=<client_name>`
    If –u or –p parameter is not set, the RACLess login panel displays; an empty value with `-p` parameter: `"-p ="`, is considered a blank password and the RACLess login panel does not display.

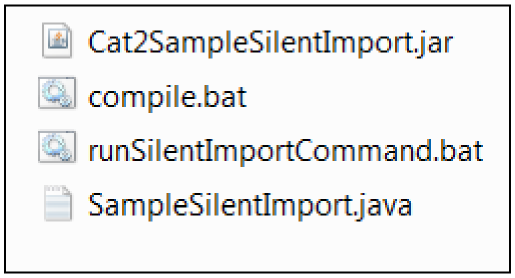
In the example below, Product1.CATProduct is defined in Data.txt file, the import process is performed on the TopLevelAssembly.CATProduct passing through SampleImportCommandLineWin.bat defined as %1 parameter.

```
SampleImportCommandLineWin.bat TopLevelAssembly.CATProduct -
u=mylogin -p=mypassword -g=dba -r=DBA -h=http://tcserver:7020/tc/ -
client=my_client_name
```

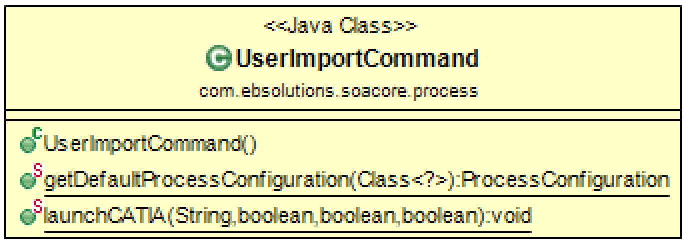
Note: Remove the `"–silent”` option from the StartRaclessServer.bat file in the SampleImportCommandLineWin.bat file to execute Import process in non-silent mode.

Method 2: Create a RACLess process that inherits from the RACLessProcessAbstract class

Refer to *%TCICV5_DIR%\custom_client\SilentImport\SampleRACLessImport\SampleSilentImport.java* class to proceed.


Cat2SampleSilentImport.jar
compile.bat
runSilentImportCommand.bat
SampleSilentImport.java

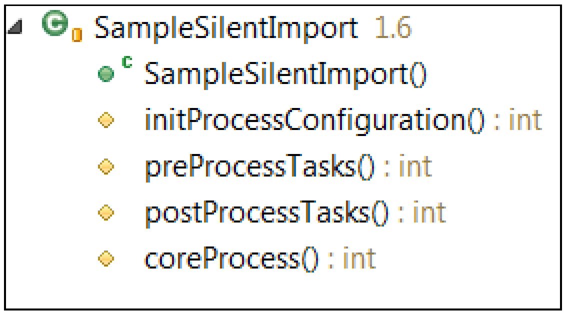
7.2.4.5.1 UserImportCommand API

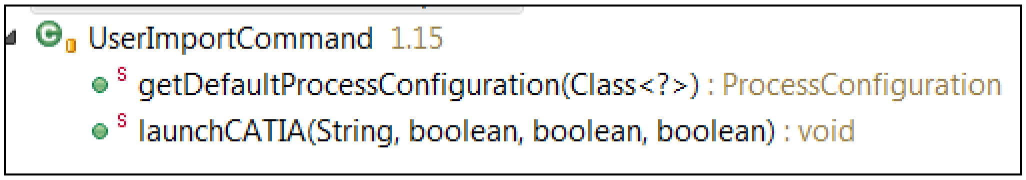

<<Java Class>>
UserImportCommand
com.ebsolutions.soacore.process

UserImportCommand()
getDefaultProcessConfiguration(Class<?>):ProcessConfiguration
launchCATIA(String,boolean,boolean,boolean):void

Do not use the default ProcessConfiguration configuration, use the API delivered in the UserImportCommand class.

7.2.4.5.2 SampleSilentImport Sample


SampleSilentImport 1.6
SampleSilentImport()
initProcessConfiguration() : int
preProcessTasks() : int
postProcessTasks() : int
coreProcess() : int


UserImportCommand 1.15
getDefaultProcessConfiguration(Class<?>) : ProcessConfiguration
launchCATIA(String, boolean, boolean, boolean) : void

This class inherits from RACLessProcessAbstract class:

- preProcessTasks() and postProcessTasks() methods are empty, by default
- initProcessConfiguration() method defines configuration through the UserExportCommand.getDefaultProcessConfiguration() API
- coreProcess() method:
    - locks the process when possible

- launches:
    - createSpreadsheet: CATScript_batch\CreateAllSpreadSheets.CATScript
    - import: CATScript_batch\ImportAllAssemblies.CATScript
- unlocks the process

Compilation and Installation of the Sample

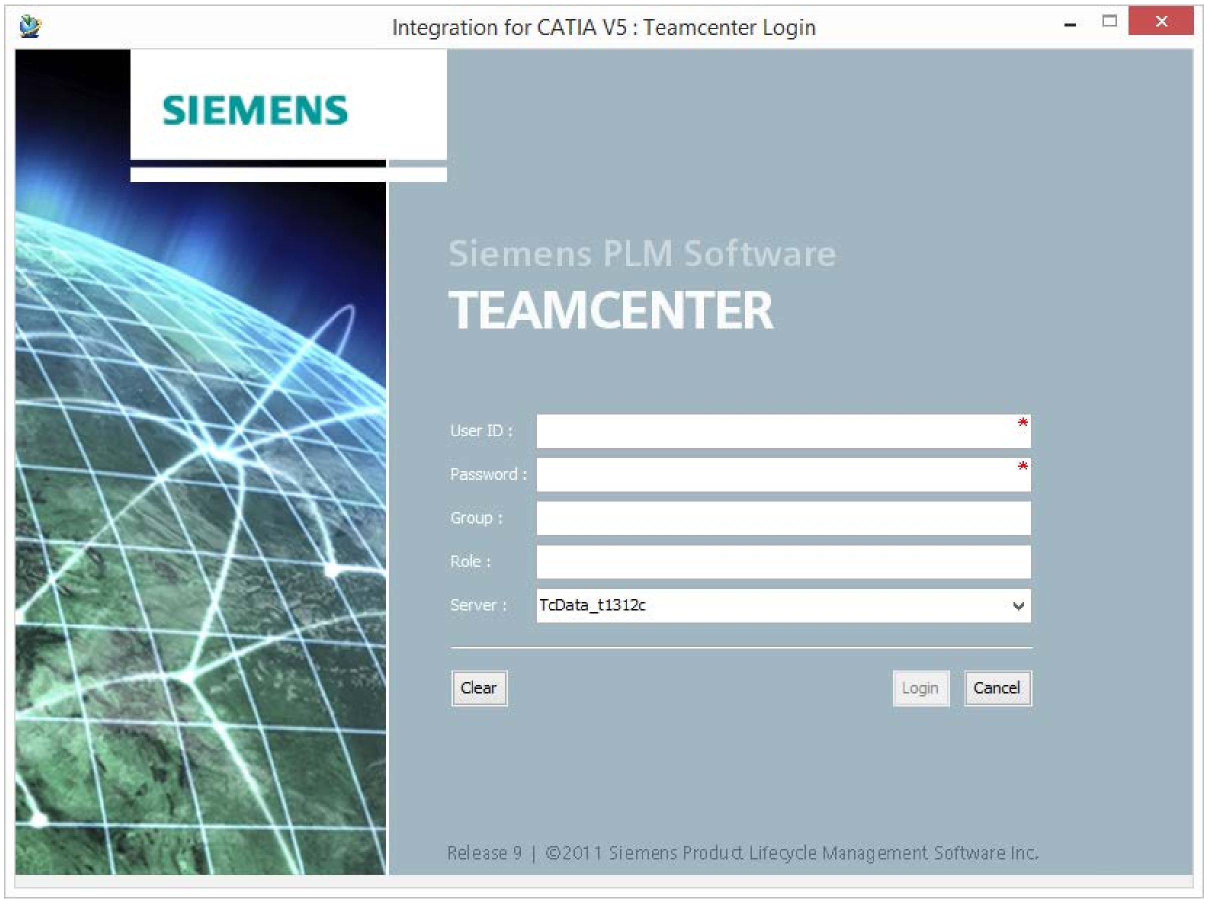
SampleSilentImport is already compiled in *%TCICV5_DIR%* so that it can be used immediately. However, the following steps must be performed to recompile and install the sample if it is necessary to customize it.

1. Locate the sample in *%TCICV5_DIR%\custom_client\silent_import\SampleRaclessImport.*
2. Update the following variables in the batch file (compile.bat):
    a. TC_ROOT
    b. FMS_HOME
    c. JAVA_HOME
    d. TCICV5_DIR
3. Modify the name of the class, if necessary.
4. Modify the name of the jar, if necessary.
5. Launch compile.bat. The compiled .jar file is now available in the current directory.
6. Modify the class path in runSilentImportCommand.bat file to *set RACLESS_CLASSPATH="%CD%\Cat2SampleSilentImport.jar"*

Execution

7. Set the **CATIA_silenced_import = true** for silent mode.
8. Run *%TCICV5_DIR%\custom_client\SilentImport\SampleRaclessImport\runSilentImportCommand.bat* with the following **optional** arguments.
    a. `-u=<user-id> -p=<password> -g=<group> -r=<role> -h=<host> -nl=<language> -proj=<project_id> -client=<client_name>`
    If –u or –p parameter is not set, the RACLess login panel displays; using an empty value with `-p` parameter: `"-p ="`, is considered as a blank password and the RACLess login panel does not display.

For example: `runSilentImportCommand.bat -u=mylogin -p=mypassword -g=dba -r=DBA -h=http://tcserver:7020/tc/ -client=my_client_name`


Integration for CATIA V5 : Teamcenter Login
SIEMENS
Siemens PLM Software
TEAMCENTER
User ID :
Password :
Group :
Role :
Server :
TcData_t1312c
Clear
Login
Cancel
Release 9 | ©2011 Siemens Product Lifecycle Management Software Inc.

If the class does not correspond to the com.user.SampleSilentImport class, then modify the following line with the correct class name:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.user.SampleSilentImport 0 11400

When the socket port has been changed, modify the following line:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.user.SampleSilentImport 0 11400

and modify the *%TCICV5_DIR%\env\tcic_var_env.xml* file:

```
<TcIC_socket_port>
11400
</TcIC_socket_port>
```

The *runSilentImportCommand.bat* can be launched.

7.2.4.6 Silent Export

UserExportCommand API

SampleSilentExport Sample

The Silent Export sample uses the tcic_selection.xml file to define the list of documents which can be exported.

Only two contexts are permitted within the *tcic_selection.xml* file for the export RACLess process; the `BOM` context, used for exporting an assembly and is equivalent to exporting from the BOM Window (with a specific configuration) and the `CATDrawing` context, used for exporting a CATDrawing and is equivalent to exporting a selected CATDrawing from My Navigator.

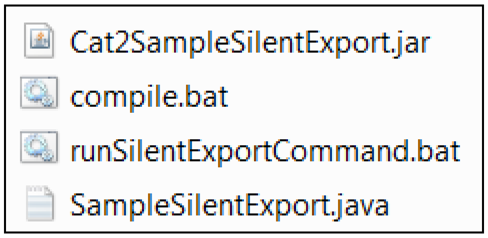
BOM example with configuration usage:

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component type="bomwindow">
    <item item_id=”A1_item_id” object_type=”A1_object_type” />
    <revision item_revision_id=”A1_item_revision_id” />
    <configuration>
          <revrule name="Working; Any Status" />
    </configuration>
    <bomline path=”A2_occ_uid/A3_occ_uid” />
</component>
<component type="bomwindow">
    <item item_id=”A10_item_id” object_type=”A10_object_type” />
    <revision item_revision_id=”A10_item_revision_id” />
    <configuration>
          <optionset name="color_blue"/>
    </configuration>
</component>
<component type="bomwindow">
    <item item_id=”A100_item_id” object_type=”A100_object_type” />
    <revision item_revision_id=”A100_item_revision_id” />
    <bomline path=”A200_occ_uid/P100_occ_uid” />
    <bomline path=”A200_occ_uid/A300_occ_uid/P200_occ_uid” />
</component>
</selection>
```

CATDrawing Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<selection>
<component uid=”existing_catdrawing_dataset_uid” />
<component uid=”existing_catdrawing_dataset_uid” />
</selection>
```

The samples for the export RACLess process are located under the *%TCICV5_DIR%\custom_client\silent_export\Windows\SampleRACless* directory.


Cat2SampleSilentExport.jar
compile.bat
runSilentExportCommand.bat
SampleSilentExport.java

There are two different methods to launch Export RACLess:

1. SampleExportCommandLineWin.bat: Does not permit customized pre/post actions and processes cannot be locked between the DoCreateExportSpreadsheetSilentAction and DoExportSilentAction commands.

2. runSilentExportCommand.bat: Permits customized pre/post actions and processes can be locked while the Create Export Spreadsheet and Export processes run.

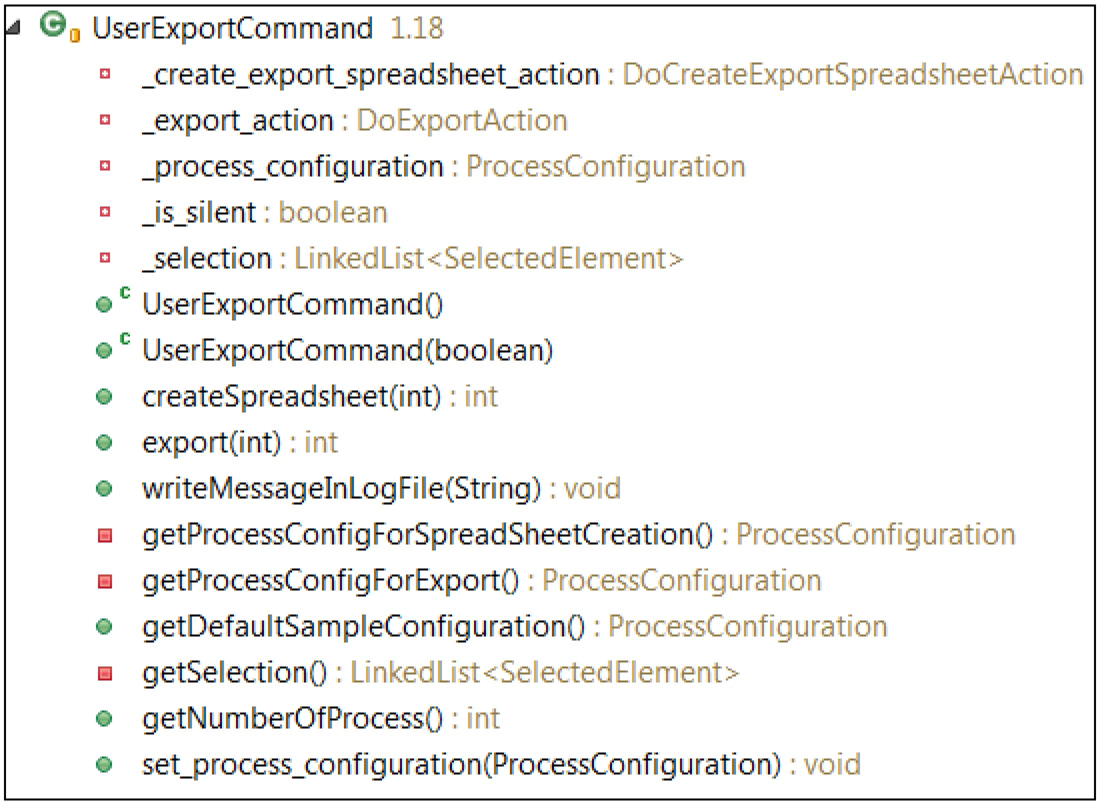
The Create Export Spreadsheet and Export processes are launched in the same server sessions. Multiple export is possible through the tcic_selection.xml file. Configurations used during Export processes is defined in the tcic_selection.xml file.

SampleExportCommandLineWin.bat launches following steps:

- StartRACLessServer.bat file in a silent or non-silent mode to launch the RACLess connection. By default, the .bat file is launched in silent mode.
- DoCreateExportSpreadsheetSilentAction command.
- DoExportSilentAction command.
- CloseRACLessServer to close the RACLess connection.

7.2.4.6.1 UserExportCommand API

Do not use default ProcessConfiguration configuration, use the API delivered in the UserExportCommand class.

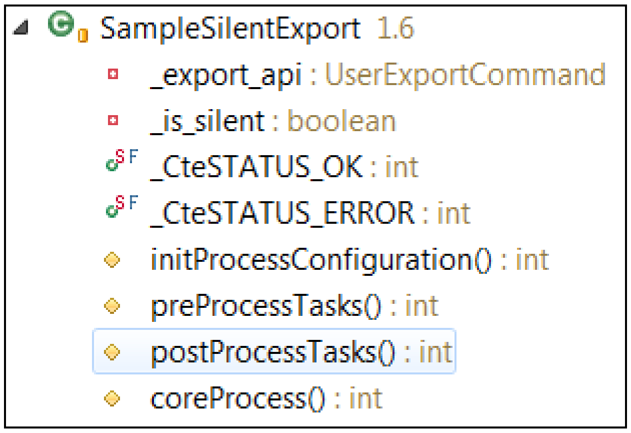


```
UserExportCommand 1.18
    _create_export_spreadsheet_action : DoCreateExportSpreadsheetAction
    _export_action : DoExportAction
    _process_configuration : ProcessConfiguration
    _is_silent : boolean
    _selection : LinkedList<SelectedElement>
    UserExportCommand()
    UserExportCommand(boolean)
    createSpreadsheet(int) : int
    export(int) : int
    writeMessageInLogFile(String) : void
    getProcessConfigForSpreadSheetCreation() : ProcessConfiguration
    getProcessConfigForExport() : ProcessConfiguration
    getDefaultSampleConfiguration() : ProcessConfiguration
    getSelection() : LinkedList<SelectedElement>
    getNumberOfProcess() : int
    set_process_configuration(ProcessConfiguration) : void
```

Use the createSpreadsheet and export functions for the export RACLess process.

The getNumberOfProcess function is used to parse the *tcic_selection.xml* file and to determine the number of documents to export.

7.2.4.6.2 SampleSilentExport Sample



```
SampleSilentExport 1.6
    _export_api : UserExportCommand
    _is_silent : boolean
    _CteSTATUS_OK : int
    _CteSTATUS_ERROR : int
    initProcessConfiguration() : int
    preProcessTasks() : int
    postProcessTasks() : int
    coreProcess() : int
```

This class inherits from RACLessProcessAbstract class:

- preProcessTasks() and postProcessTasks() methods are empty, by default
- initProcessConfiguration() method defines configuration through the UserExportCommand.getDefaultProcessConfiguration() API and sets the silent or non-silent mode.
- coreProcess() method:
    - locks the process when possible
    - reads the *tcic_selection.xml* file
    - loops on each valid document defined in the query.xml file and calls for each document: createSpreadsheet and export
    - unlocks the process

Compilation and Installation of the Sample

Perform the following steps to compile and install the sample.

1. Locate the sample in *%TCICV5_DIR%\custom_client\silent_export\Windows\SampleRacless.*
2. Update the following variables in the batch file (compile.bat):
    a. TC_ROOT
    b. FMS_HOME
    c. JAVA_HOME
    d. TCICV5_DIR
3. Modify the name of the class, if necessary.
4. Modify the name of the jar, if necessary.
5. Launch compile.bat. The compiled .jar file is now available in the current directory.

Install the .jar file

The .jar file can be installed under *%TCICV5_DIR%\racless\extensions* directory. In this case, the jar is automatically added to the TcIC classpath.

The .jar file can also be installed under another directory. Add the new directory to the TcIC class path.

6. For example, the *Cat2SampleSilentExport.jar* file is compiled under the *F:\Temp* directory. Modify the *%TCICV5_DIR%\racless\StartRaclessServer.bat* file as follows:

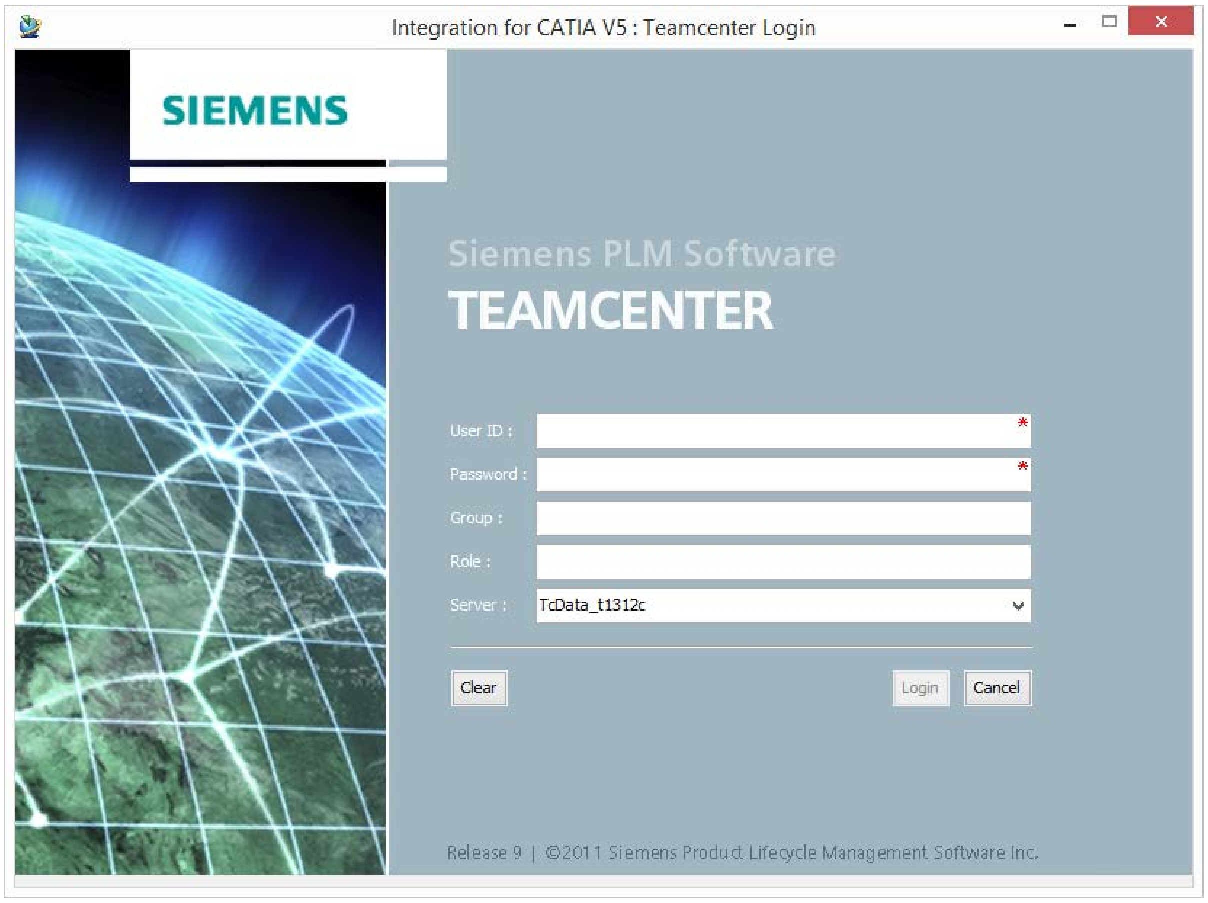
   a. Add the .jar file to the classpath RACLESS_CLASSPATH
      set RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fscjavaclientproxy.jar
      set RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fccjavaclientproxy.jar
      set RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fmsutil.jar
      set RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;F:\Temp\Cat2SampleSilentExport.jar
      FOR /R "%RACLESS_DIR%" %%a in (*.jar) DO CALL :AddToPath "%%a"

Execution

7. Run *%TCICV5_DIR%\custom_client\silent_export\Windows\SampleRACLess\runSilentExportCommand.bat* with the following **optional** arguments:

   a. `-u=<user-id> -p=<password> -g=<group> -r=<role> -h=<host> -nl=<language> -proj=<project_id> -client=<client_name>`
      If –u or –p parameter is not set, the RACLess login panel displays; an empty value with `-p` parameter: `"-p ="`, is considered a blank password and the RACLess login panel does not display.

For example, `runSilentExportCommand.bat -u=mylogin -p=mypassword -g=dba -r=DBA -h=http://tcserver:7020/tc/ -client=my_client_name`


Integration for CATIA V5 : Teamcenter Login
SIEMENS
Siemens PLM Software
TEAMCENTER
User ID :
Password :
Group :
Role :
Server :
TcData_t1312c
Clear
Login
Cancel
Release 9 | ©2011 Siemens Product Lifecycle Management Software Inc.

If the class does not correspond to the com.user.SampleSilentExport class, then modify the following line with the correct class name:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.user.SampleSilentExport 0 11400

When the socket port has been changed, modify the following line:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.user.SampleSilentExport 0 11400

And modify the *%TCICV5_DIR%\env\tcic_var_env.xml* file:

```
<TcIC_socket_port>
11400
</TcIC_socket_port>
```

The *tcic_selection.xml* file can now be created and r*unSilentExportCommand.bat* can be launched.

7.2.4.7 Silent Load, Update, Save

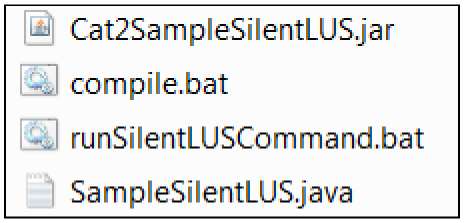
UserLUSCommand API
SampleSilentLUS Sample

The Silent Load, Update, Save (LUS) sample uses the tcic_selection.xml file to define the list of documents which can be loaded and saved.

All contexts are permitted in the tcic_selection.xml file for the LUS RACLess processes.

The Load, Update, Save process on multiple BOM windows is possible through the tcic_selection.xml file.

The samples for LUS RACLess processes are located under *custom_client\Silent_lus\Windows\SampleRacless.*


Cat2SampleSilentLUS.jar
compile.bat
runSilentLUSCommand.bat
SampleSilentLUS.java

There are two different methods to launch LUS RACLess:

1. SampleLUSCommandLineWin.bat: Does not permit customized pre/post actions or launching the DoLUSSilentAction command.

2. runSilentLUSCommand.bat: Permits customized pre/post actions and processes can be locked while the LUS process runs.

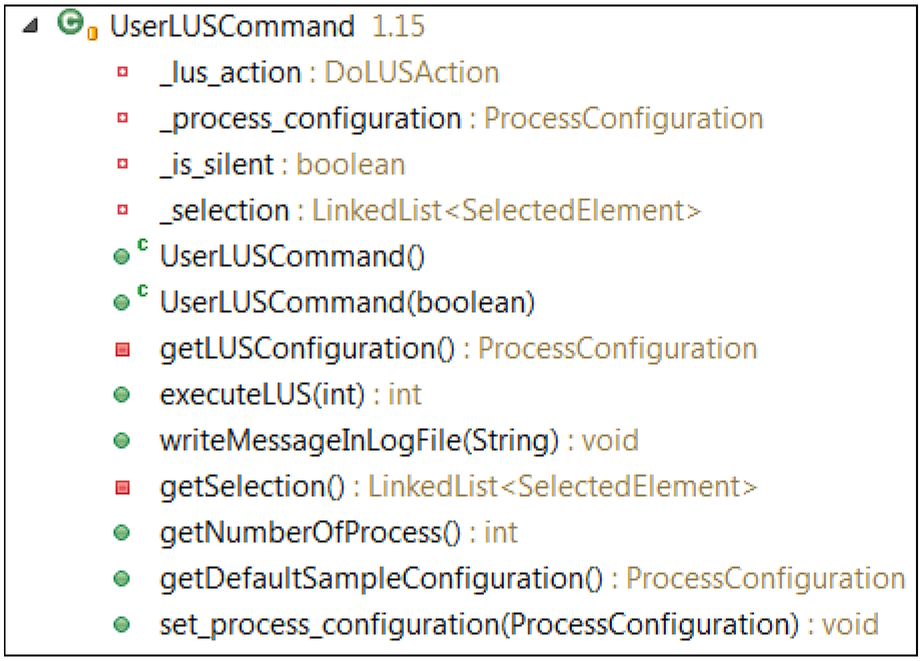
SampleLUSCommandLineWin.bat launches following steps:

- StartRACLessServer.bat file in a silent or non-silent mode to launch the RACLess connection. By default, the .bat file is launched in silent mode.

- DoLUSSilentAction command

- CloseRACLessServer to close the RACLess connection.

7.2.4.7.1 UserLUSCommand API

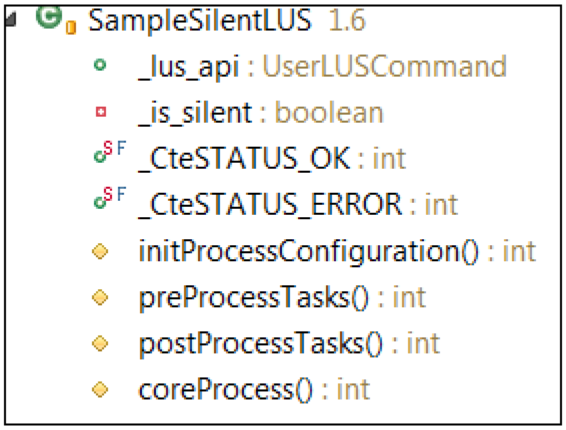


```
UserLUSCommand 1.15
    _lus_action : DoLUSAction
    _process_configuration : ProcessConfiguration
    _is_silent : boolean
    _selection : LinkedList<SelectedElement>
    UserLUSCommand()
    UserLUSCommand(boolean)
    getLUSConfiguration() : ProcessConfiguration
    executeLUS(int) : int
    writeMessageInLogFile(String) : void
    getSelection() : LinkedList<SelectedElement>
    getNumberOfProcess() : int
    getDefaultSampleConfiguration() : ProcessConfiguration
    set_process_configuration(ProcessConfiguration) : void
```

Do not use default ProcessConfiguration configuration, use the API delivered in the UserLUSCommand class.

The `executeLUS` function manages all documents defined in the *tcic_selection.xml* file or all documents set in ModelObject Vector by `set_documents` function.

7.2.4.7.2 SampleSilentLUS Sample



```
SampleSilentLUS 1.6
    _lus_api : UserLUSCommand
    _is_silent : boolean
    _CteSTATUS_OK : int
    _CteSTATUS_ERROR : int
    initProcessConfiguration() : int
    preProcessTasks() : int
    postProcessTasks() : int
    coreProcess() : int
```

This class inherits from RACLessProcessAbstract class:

- preProcessTasks() and postProcessTasks() methods are empty, by default
- initProcessConfiguration() method defines configuration through the UserLUSMCommand.getDefaultProcessConfiguration() API and sets the silent or non-silent mode.
- coreProcess() method:
  - locks the process when possible
  - instantiates the UserLUSCommand with the query.xml file date
  - launches LUS
  - unlocks the process.

Compilation and Installation of the Sample

Perform the following steps to compile and install the sample.

1. Locate the sample in *%TCICV5_DIR%\custom_client\Silent_lus\Windows\SampleRacless.*
2. Update the following variables in the batch file (compile.bat):
   a. TC_ROOT
   b. FMS_HOME
   c. JAVA_HOME
   d. TCICV5_DIR
3. Modify the name of the class, if necessary.
4. Modify the name of the jar, if necessary.
5. Launch compile.bat. The compiled .jar file is now available in the current directory.

Install the .jar file

The .jar file can be installed under *%TCICV5_DIR%\racless\extensions* directory. In this case, the jar is automatically added to the TcIC classpath.

The .jar file can also be installed under another directory. In this case, add this new directory to the TcIC class path.

6. For example, the *Cat2SampleSilentLUS.jar* file is compiled under the *F:\Temp* directory. Modify the *%TCICV5_DIR%\racless\StartRaclessServer.bat* file as follows:

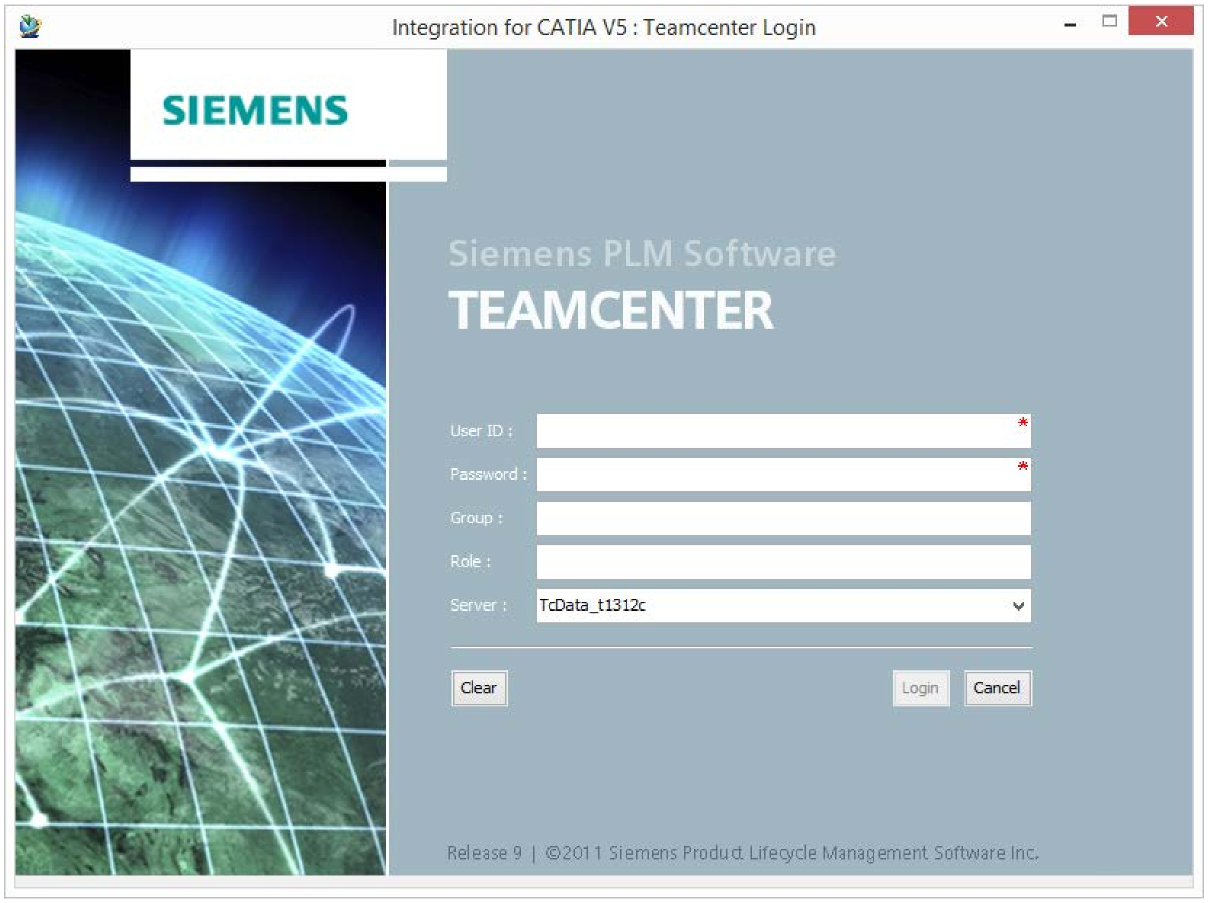
   a. Add the .jar file to the classpath RACLESS_CLASSPATH
      set
      RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fscjavacli
      entproxy.jar
      set
      RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fccjavacli
      entproxy.jar
      set
      RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;%FMS_HOME%\jar\fmsutil.jar
      set
      RACLESS_CLASSPATH=%RACLESS_CLASSPATH%;F:\Temp\Cat2SampleSilentLU
      S.jar
      FOR /R "%RACLESS_DIR%" %%a in (*.jar) DO CALL :AddToPath "%%a"

Execution

7. Run *%TCICV5_DIR%\custom_client\silent_lus\Windows\SampleRACLess\runSilentLUSCommand.bat* with the following **optional** arguments:

   a. `-u=<user-id> -p=<password> -g=<group> -r=<role> -h=<host> -nl=<language> -proj=<project_id> -client=<client_name>`
      If –u or –p parameter is not set, the RACLess login panel displays; an empty value with `-p` parameter: `"-p ="`, is considered a blank password and the RACLess login panel does not display.

For example: `runSilentLUSCommand.bat -u=mylogin -p=mypassword -g=dba -r=DBA -h=http://tcserver:7020/tc/ -client=my_client_name`


Integration for CATIA V5 : Teamcenter Login
SIEMENS
Siemens PLM Software
TEAMCENTER
User ID :
Password :
Group :
Role :
Server :
TcData_t1312c
Clear
Login
Cancel
Release 9 | ©2011 Siemens Product Lifecycle Management Software Inc.

If the class does not correspond to the com.user.SampleSilentLUS class, then modify the following line with the correct class name:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" **com.user.SampleSilentLUS** *0 11400*

When the socket port has been changed, modify the following line:

"%TCICV5_DIR%\bin\WIN32\socketV5.exe" com.user.SampleSilentLUS 0 **11400**

and modify the *%TCICV5_DIR%\env\tcic_var_env.xml* file:

```
<TcIC_socket_port>
11400
</TcIC_socket_port>
```

The *tcic_selections.xml* file can now be created and *runSilentLUSCommand.bat* can be launched.

7.2.4.8 RACLess API JAVA Documentation

Refer to the online documentation for access to these files.

7.2.5 RACLess Command Line Entry Points

The Integration provides the ability to run some of the RACLess processes used within the Teamcenter Integration for CATIA V5 directly under a command line.

This chapter contains methods to call the RACLess processes used within the Teamcenter Integration for CATIA V5.

Important Information Related to RACLess and Command Lines

The **DoLoginRACLessAction** command provides the ability to connect to the RACLess application with an external customized client application, such as Custom Loader, instead of the portal client.

An external client application or Custom Loader must be defined in the client_env file using **Client_1** in the Category line to indicate the client is a Custom Loader.

1. Launch `StartRACLessServer.bat` using the appropriate parameter to set the client parameters.

2. Launch `DoLoginRACLessAction` in a command line using the appropriate credentials to establish the client connection when the first command is launched.

Parameter credentials are set through the `DoLoginRACLessAction` command when the first command launches.

For all processes, ensure the tcic_selection.xml file exists before launching the command, except DoLoginRACLessAction and DoUserSettingsAction.

The 11400 value corresponds to the value of the socket port defined for the TcIC_socket_port variable in the tcic_var_env.xml file, except DoLoginRACLessAction.

The **socketV5.exe** parameter controls the synchronous vs. asynchronous operations of TcIC commands, i.e., CATIA may be locked until the command finishes (value is **0**) or remain unlocked (value is **1**), depending on the particular command launched from a cmd line.

The best practice is to lock the session, indicated with the value **0** (default). CATIA should remain unlocked (**1**) when using AW as a client.

Process	Available Customization
Browse Multi-model Links	View multi-model links of a document and its related attributes in Teamcenter. This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset or form information. `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoBMMLAction 0 11400`

Check In/Check Out menu in Teamcenter	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark.
	
	Check In:
	
	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoCheckInAction 0 11400`
	
	Check Out:
	
	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoCheckOutAction 0 11400`

Process	Available Customization

Connection to RACLess Application	External client:
	
	Share the custom loader credentials to the RACLess application using the command line:
	
	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoginRACLessAction 0 11400 –client=Loader –u=user_id –h=host`
	
	List of parameters:
	
	`-client=<client_name> (required)`
	`-u=<user_id> (optional)`
	`-p=<password> (optional)`
	`-g=<group> (optional)`
	`-r=<role> (optional)`
	`-h=<host> (optional)`
	`-nl=<lang> ( optional)`
	`-u=<user_id> (optional)`
	`-proj=<project_id> (optional)`

Create CATPart dataset for selection	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, ProductContextInfo or bookmark. Create a CATPart dataset within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoCreateCATPartDatasetAction 0 11400`
Create CATProduct dataset for selection	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, ProductContextInfo or bookmark. Create a CATProduct dataset within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoCreateCATProductDatasetAction 0 11400`
Create Export Spreadsheet	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Create Export Spreadsheet within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoCreateExportSpreadsheetAction 0 11400`

Process	Available Customization
Export	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Export a document from Teamcenter in CATIA within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoExportAction 0 11400`
Export as reference	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Export a document from Teamcenter in CATIA within a command line for reference: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoExportAsReferenceAction 0 11400`
Export to directory	Use Export to directory command, available on product, part, drawing, process, analysis, desing table, foreign file or catalog item, item revision, dataset, bom line, to select directory in which selected documents are exported: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoExportToDirectoryAction 0 11400 –d=directory` `-d=<directory >: directory in which selected documents are exported` (optional) When directory is not defined, a directory browser is displayed.
Highlight in CATIA	This command uses a *tcic_selection.xml* file containing a defined BOM window, ProductContextInfo or bookmark. `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoHighlightInCatiaAction 0 11400`
Load	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Load a document from Teamcenter in CATIA within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadAction 0 11400`

Process	Available Customization
Load as reference	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Load a document from Teamcenter in CATIA within a command line for reference: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadAsReferenceAction 0 11400`
Load for Compare	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Load a document from Teamcenter in CATIA within a command line for comparison: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadForCompareAction 0 11400`
Load for Consulting	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Load a document from Teamcenter in CATIA within a command line for consulting : `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadForConsultingAction 0 11400`
Load Manager	Display the Load Manager panel to determine options and command to launch in the Open in CATIA command form Active Workspace: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoDisplayLoadManagerAction 0 11400`
Load, Update, Save	Load Update Save a document from Teamcenter in CATIA within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLUSAction 0 11400`
Teamcenter Loader	Display Teamcenter Loader Panel with the command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoDisplayLoaderAction 0 11400`
Open dataset	This command uses a *tcic_selection.xml* file containing a defined dataset. Open a dataset document from Teamcenter in CATIA `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoOpenDatasetAction 0 11400`

Process	Available Customization
Open in CATIA from Active Workspace	These commands use a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. **DoLoadAction** (default) launches when launching Open in CATIA commands when Load Manager is not displayed. **DoLoadFromLauncherAction** command is taken into account the current CATIA command to determine which type of Load command should be launched from Active Workspace inside CATIA: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadFromLauncherAction 0 11400` By default, **DoLoadFromLauncherAction** command is used if no specific command is defined in *TcICLauncher.bat*. **DoDefaultLoadFromLauncherAction** command is taken into account the current CATIA command to determine which type of Load command should be launched with the following configuration **CATIA_load_bom_traversal_option** set to **THROUGH**, **CATIA_load_as_cgr set** to **false**, **CATIA_activate_latestActiveShapes** set to **1**, **CATIA_export_only_active_shapes** set to **0** , from Active Workspace inside CATIA: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoDefaultLoadFromLauncherAction 0 11400` **DoLoadWithOptionPanelAction** command displays Load Manager panel when Open in CATIA is launched: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoLoadWithOptionPanelAction 0 11400`

Process	Available Customization

Save Assembly Configuration	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision information and configuration to use for BOM window creation, ProductContextInfo or bookmark.
	
	Save Assembly configuration file used for the Load as reference process in non-silent mode:
	
	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSaveAssemblyConfigurationAction 0 11400 -dstuid=uid  -dstid=id -dstrev=rev -dstname=name -dstdesc=desc -filename=name`
	
	Save Assembly configuration file used for the Load as reference process in silent mode:
	
	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSilentSaveAssemblyConfigurationAction 0 11400 -dstuid=uid  -dstid=id -dstrev=rev -dstname=name -dstdesc=desc -filename=name`
	
	List of parameters:
	
	`-dstuid=<uid >: Dataset uid` (optional)
	`-dstid=<id> : Dataset identifier` (optional)
	`-dstrev=<rev >: Dataset revision` (optional)
	`-dstname=<name> : Dataset name` (optional)
	`-dstdesc=<desc >: Dataset description` (optional)
	`-filename=<name >: File name` (optional)

Silent Create Export Spreadsheet	Create Export Spreadsheet in silent mode within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSilentCreateExportSpreadsheetAction 0 11400`
Silent Export	Export a document in a Silent mode from Teamcenter in CATIA within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSilentExportAction 0 11400`

Process	Available Customization
Silent Export to directory	Use Export to directory command in a silent mode , available on product, part, drawing, process, analysis, desing table, foreign file or catalog item, item revision, dataset, bom line, to select directory in which selected documents are exported: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSilentExportToDirectoryAction 0 11400 –d=directory` `-d=<directory>: directory in which selected documents are exported` (optional)
Silent Load, Update, Save	This command uses a *tcic_selection.xml* file containing a defined BOM window, item revision, dataset, ProductContextInfo or bookmark. Load Update Save a document in a Silent mode from Teamcenter in CATIA within a command line: `"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoSilentLUSAction 0 11400`

User Settings	`"%TCICV5_DIR%\bin\WIN32\socketV5.exe" DoUserSettingsAction 0 11400 -g=group -r=role -proj=project_id`
	
	List of parameters:
	
	`-g=<group> (optional)`
	`-r=<role> (optional)`
	`-proj=<project_id> (optional)`
	
	The RACLess connection is not modified when parameters are not defined.
	
	The command is not available when Teamcenter RAC Client is running.
	
	If `–proj="`, then the session launches without the project’s configuration.

7.3 Save Manager Developer Guide

7.3.1 Overview

The Save Manager Developer Guide contains methods, class hierarchies and steps required to customize the Save Manager used within the Teamcenter Integration for CATIA V5. The Save Manager collects information needed during the Save process and displays when a CATIA user performs a Save operation on an assembly in Teamcenter, using the Integration Save macros.

Use the samples described in this document as the starting point for your specific customizations. The Save Manager interface has been designed with accessible graphic elements and CATIA datas for customization.

Contact supportEngineering@ebsolutions.fr for further assistance with customizations, if necessary.

7.3.2 Save Manager Configuration File

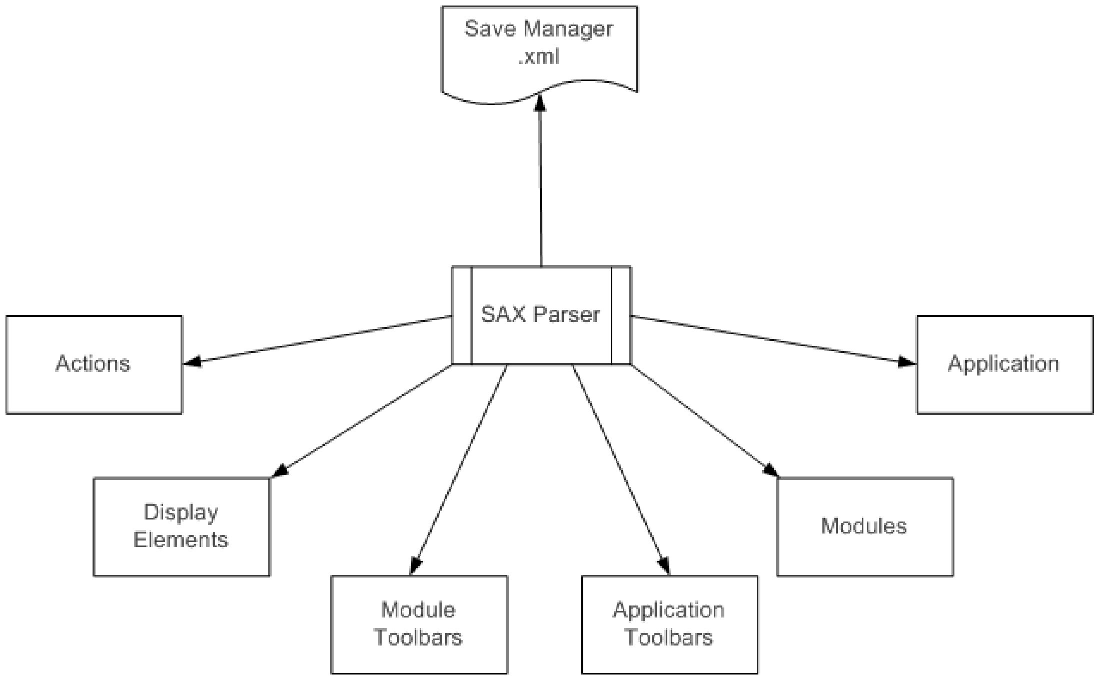
The savemanager.xml file is the entry point for every display object, as illustrated below. The .xml file defines elements within the Save Manager panel, such as application, modules, toolbars, display elements and actions associated to the display elements. The SAX parser retrieves the .xml file information.

The savemanager.xml file uses pre-defined schemas to display the Save Manager, and has two levels of customization:

1. **High Level** customization occurs when modifications to pre-defined options within the .xml file are performed. These pre-defined options are managed by the "out of the box" Integration. Refer to the Customizing the Save Manager for information on high level .xml customizations.

2. **Low Level** customization uses Java components created by administrators to integrate it into the Save Manager or change the behavior of the components. A new code signature

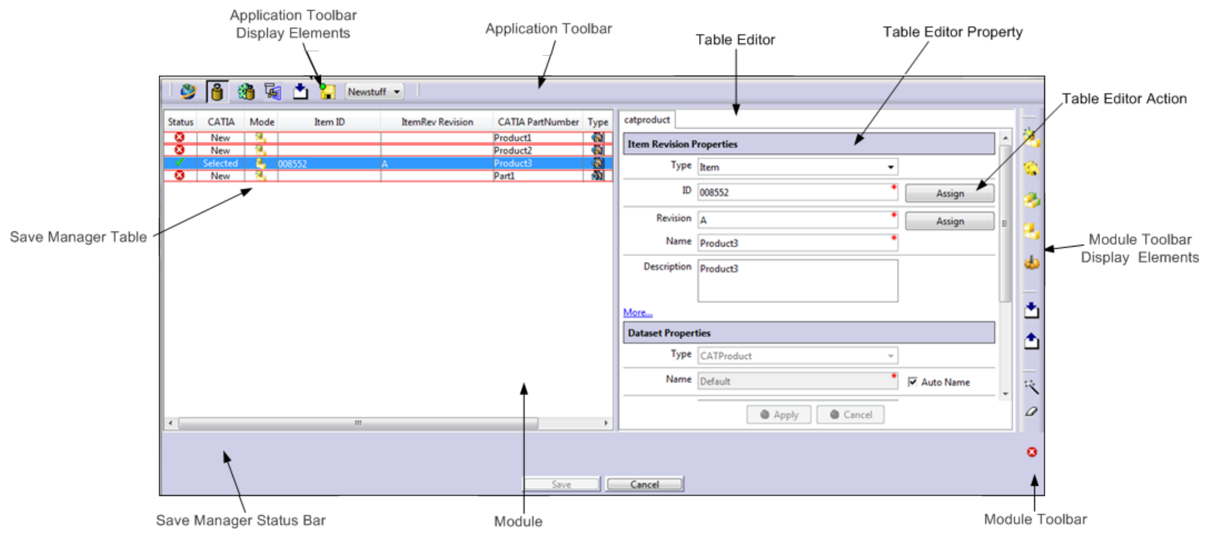
definition (`class = new class`) is added to the .xml file, and the Factories Mechanism instantiates the .xml class definitions.



7.3.2.1 Save Manager Configuration File: Graphical Objects Description

The Save Manager interface is composed of several customizable components, as illustrated below.

The Module area includes the display between the top and left toolbars, including the table and the table editor. Refer to the Create a New Module section to customize this area.



Status	CATIA	Mode	Item ID	ItemRev Revision	CATIA PartNumber	Type
✗	New				Product1	
✗	New				Product2	
✓	Selected		008552	A	Product3	
✗	New				Part1	

Type: Item
ID: 008552
Revision: A
Name: Product3
Description: Product3

Type: CATProduct
Name: Default
☑ Auto Name

7.3.2.2 Save Manager Configuration File: XML Tags

[Definitions](#)

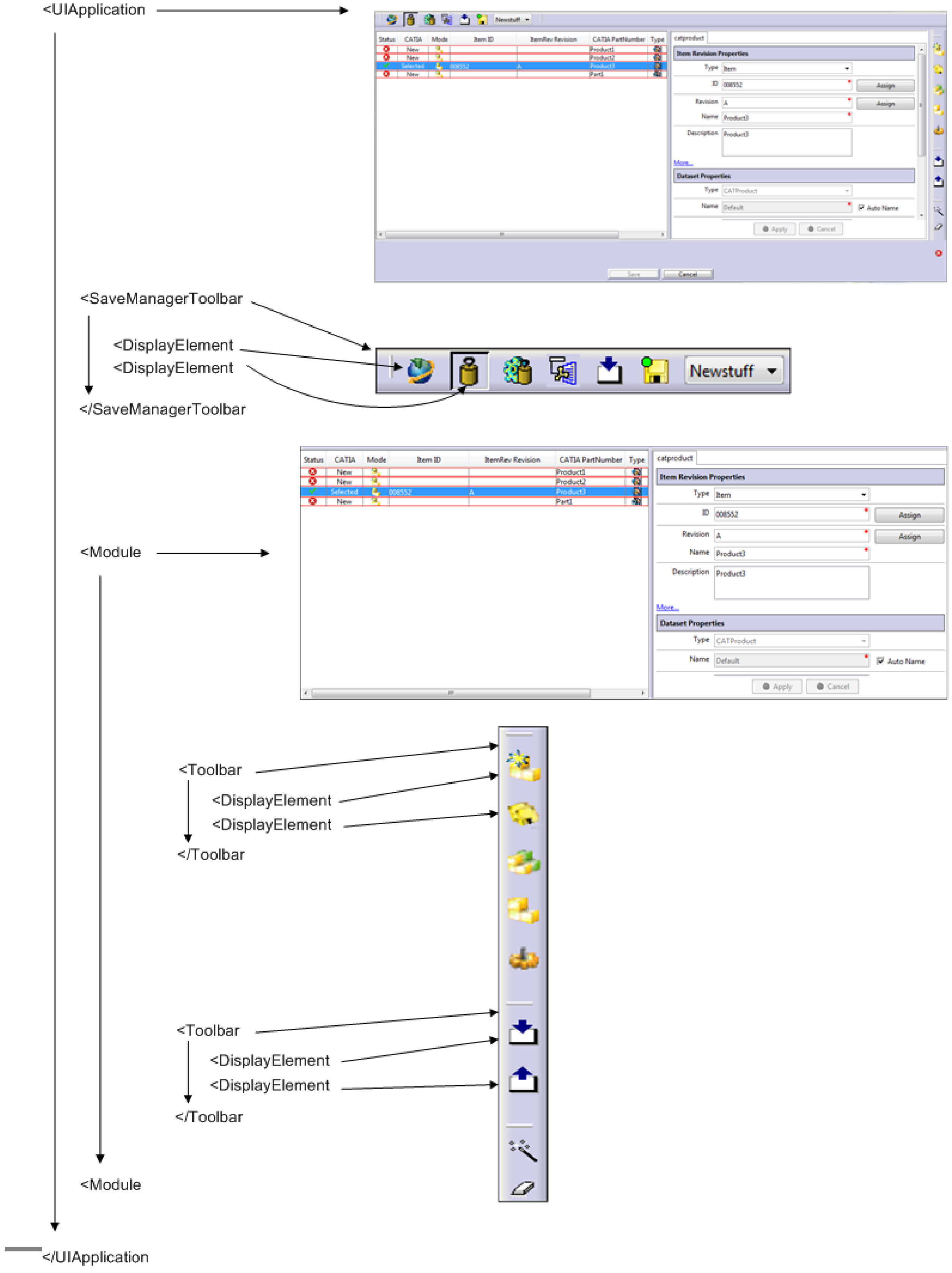
[Names and Descriptions](#)

[Hierarchy](#)

7.3.2.2.1 Definitions

Use the *<TCICV5_DIR>/env/savemanager.xml* file provided with the installation to control the appearance and behavior of the Save Manager. Refer to the illustration below for definitions of each xml display tag within the *savemanager.xml* file.

The default Save Manager includes only one module (the Save Module) but it can manage more than one module. If more than one module is defined in the xml file, a side bar on the left side of the Save Manager displays as many buttons as modules defined. Display the module by picking the button representing that module.



Status	CATIA	Mode	Item ID	ItemRev Revision	CATIA PartNumber	Type
	New				Product1	
	New				Product2	
	Selected		008552	A	Product3	
	New				Part1	

catproduct

Item Revision Properties

Type: Item
ID: 008552 Assign
Revision: A Assign
Name: Product3
Description: Product3

More...

Dataset Properties

Type: CATProduct
Name: Default ☑ Auto Name

Apply Cancel

7.3.2.2.2 Names and Descriptions

Refer to the following chart for tag names and their description.

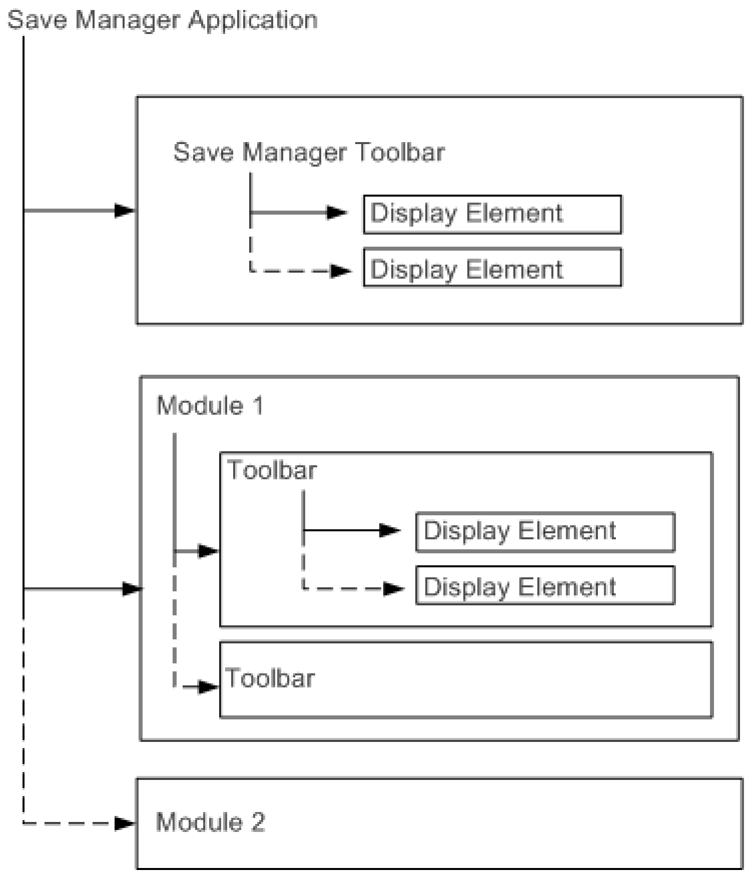
If an element has multiple definitions, ensure the **name**, **nameRef**, or **identifier** attributes of the new element are changed. If two elements have the same name, only one appears because the Integration considers it unique depending on its key.

Tag Name	Possible Parents Tags	Possible Children Tags	Multiple Definitions
`UIApplication`	No Parent : UIApplication is the root Tag	SaveManagerToolBar,Module	No
`SaveManagerToolBar`	UIApplication	DisplayElement	Yes
`Module`	UIAapplication	ToolBar	Yes
`ToolBar`	Module	DisplayElement	Yes
`DisplayElement`	SaveManagerToolBar,ToolBar	No child	Yes

7.3.2.2.3 Hierarchy

Each tag in the *savemanager.xml* file contains a specific hierarchy, as displayed below.

Modifying the tag hierarchy may cause required definitions to be bypassed, and prevent the Save Manager from being displayed.



7.3.3 Save Manager Entry Point

IEntryPointForButtonAction defines extension points that are called for pre- and post-actions associated to a button of the Save Manager or Spreadsheet panel.

This interface can have several implementations. Each implementation is automatically called when defined in the *tcic_extensions.xm*l file.

Implement the **userEntryPreButtonAction** and **userEntryPostButtonAction** method to define your own behavior. Refer to the User Interface Implementation: Main Classes section for details on extension point implementations, if necessary.

7.3.3.1 Customization Examples

Extensions to be called are defined in the *tcic_extensions.xml* file.

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

```
   <entry
       id="com.ebsolutions.savemanager.extension.IEntryPointForButt
onAction"
       filename="EntryPointsImplForSM.jar"
       handler="IEntryPointForButtonActionImpl1">
   </entry>
   <entry
       id="com.ebsolutions.soacore.extensions.ICatiaUserExit"
       filename="EntryPointsImplForSM.jar"
       handler="CatiaUserExitTest1">
   </entry>
</extensions>
```

Implement IEntryPointForButtonActionImpl1 to define your own behavior. Traces display in the console when the user clicks on the buttons.

```
public class IEntryPointForButtonActionImpl1 implements
IEntryPointForButtonAction

{

 @Override

public boolean userEntryPostButtonAction(String
arg0,SaveManagerApplication arg1);

   {

       System.out.println("jtrace -->> userEntryPostButtonAction :
arg0 = " + arg0 + " - arg1 = " + arg1);

       return true;

   }

   @Override

public boolean userEntryPreButtonAction(String
arg0,SaveManagerApplication arg1);

   {

       System.out.println("trace -->> userEntryPreButtonAction :
arg0 = " + arg0 + " - arg1 = " + arg1);

       return true;

   }

}
```

7.3.4 Save Manager Lifecycle

The Save Manager lifecycle begins with the .xml file and ends with data manipulation. The "out of the box" Save Manager is delivered with a powerful, customizable xml engine to meet business needs, and has easy data access. Use **Handlers** and **Interfaces** to customize the Save Manager behavior and design.

7.3.4.1 CATIAdatas

Review the following sections to determine how the CATIA structure information is dispatched in the Save Manager.

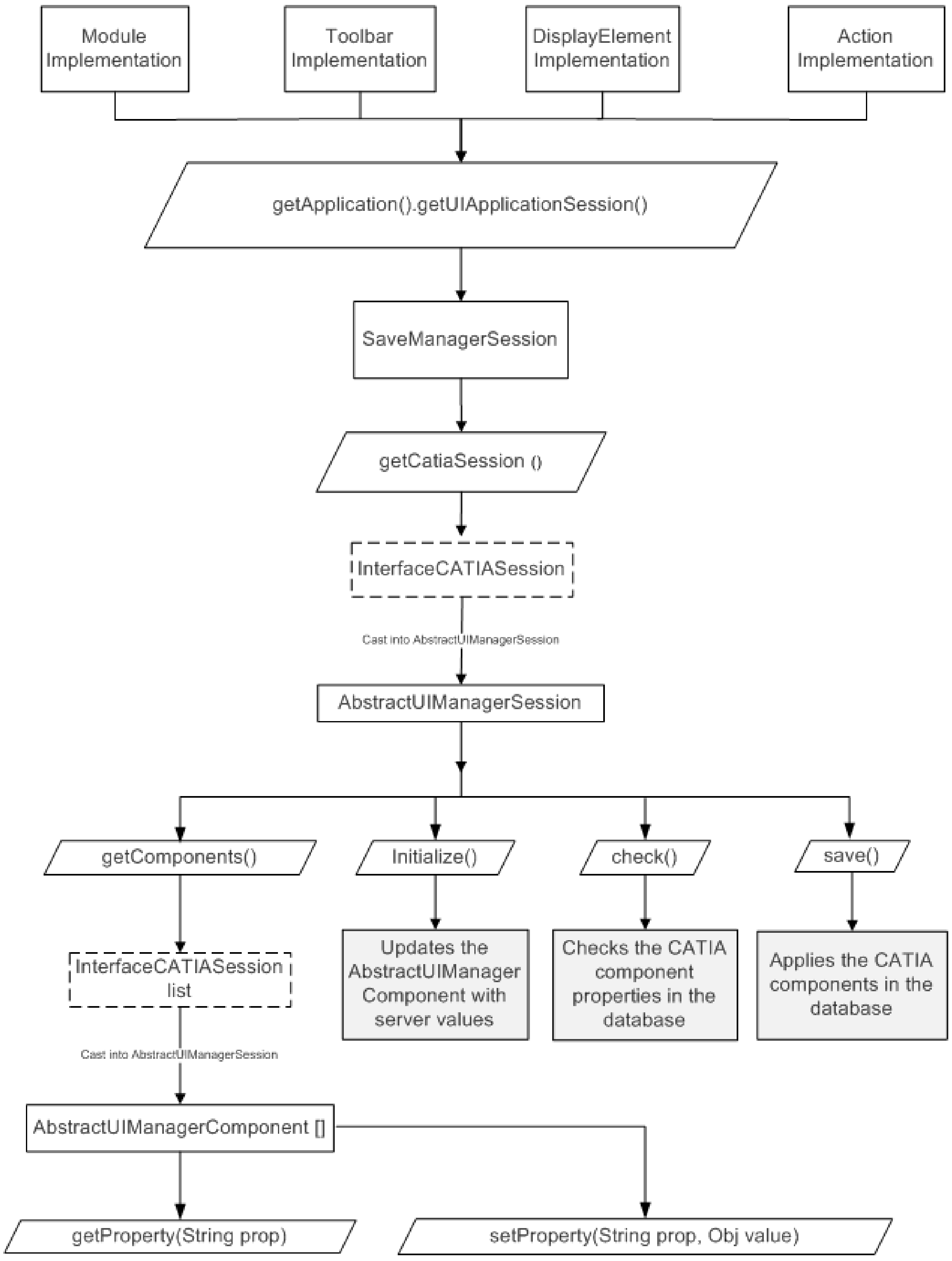
7.3.4.1.1 Accessing CATIAdatas

The Save Manager is designed to access the CATIA structure from everywhere using the **getApplication().getUIApplicationSession()** methods. The SaveManagerSession is the gate between the graphics elements and CATIAdatas and is populated when created with an InterfaceAbstractUIManagerSession.

The **InterfaceAbstractUIManagerSession** is an interface used to manipulate the CATIA session and components representation. When the SaveManagerSession class is created, the Integration interface provides the AbstractUIManagerSession object containing all the CATIA structure information. This AbstractUIManagerSession object implements the InterfaceAbstractUIManagerSession allowing it to pass to the SaveManagerSession.

Manipulate CATIA data from the AbstractUIManagerSession object. The getComponents() method provides the AbstractUIManagerComponent that exists in CATIA. From any component, every property of the CATIA document can be accessed using the getProperty(String propName) method or to set any property using the setProperty(String propName, obj value) method.

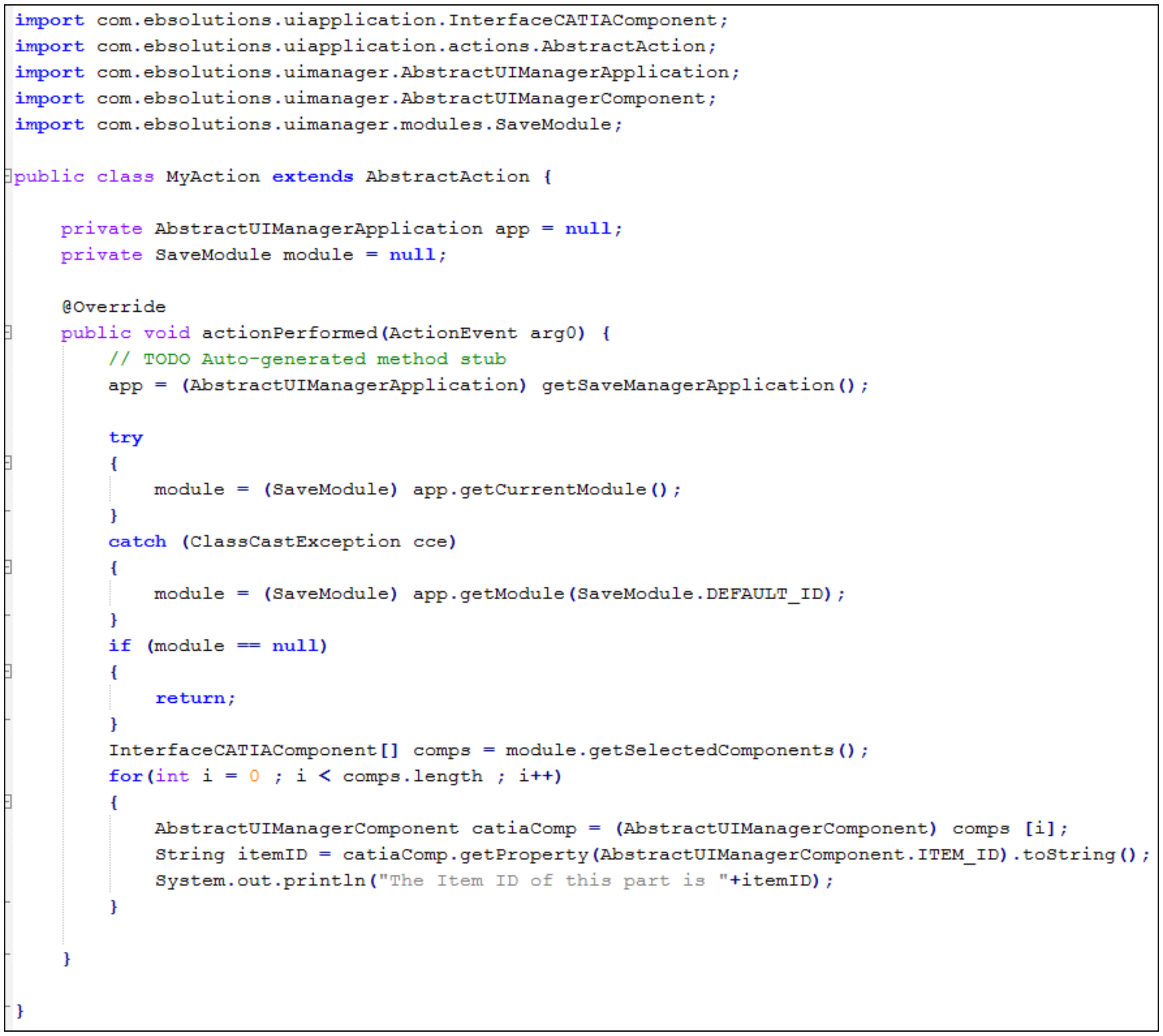
Accessing the SaveManagerSession from an Implementation


Module Implementation
Toolbar Implementation
DisplayElement Implementation
Action Implementation
getApplication().getUIApplicationSession()
SaveManagerSession
getCatiaSession ()
InterfaceCATIASession
Cast into AbstractUIManagerSession
AbstractUIManagerSession
getComponents()
Initialize()
check()
save()
InterfaceCATIASession list
Updates the AbstractUIManager Component with server values
Checks the CATIA component properties in the database
Applies the CATIA components in the database
Cast into AbstractUIManagerSession
AbstractUIManagerComponent []
getProperty(String prop)
setProperty(String prop, Obj value)

7.3.4.1.2 Accessing Selected Components

Use the following steps to retrieve a list of selected components in the Save Manager Table using an action implementation.

1. Retrieve the application using the getSaveManagerApplication() method.
2. From the application, use the getCurrentModule() method to get the SaveModule module (the SaveModule determines which object is selected).
3. Use the getSelectedComponenets() method to get all the AbstractUIManagerComponents selected in the Save Module Table.



```
import com.ebsolutions.uiapplication.InterfaceCATIAComponent;
import com.ebsolutions.uiapplication.actions.AbstractAction;
import com.ebsolutions.uimanager.AbstractUIManagerApplication;
import com.ebsolutions.uimanager.AbstractUIManagerComponent;
import com.ebsolutions.uimanager.modules.SaveModule;

public class MyAction extends AbstractAction {

    private AbstractUIManagerApplication app = null;
    private SaveModule module = null;

    @Override
    public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub
        app = (AbstractUIManagerApplication) getSaveManagerApplication();

        try
        {
            module = (SaveModule) app.getCurrentModule();
        }
        catch (ClassCastException cce)
        {
            module = (SaveModule) app.getModule(SaveModule.DEFAULT_ID);
        }
        if (module == null)
        {
            return;
        }
        InterfaceCATIAComponent[] comps = module.getSelectedComponents();
        for(int i = 0 ; i < comps.length ; i++)
        {
            AbstractUIManagerComponent catiaComp = (AbstractUIManagerComponent) comps [i];
            String itemID = catiaComp.getProperty(AbstractUIManagerComponent.ITEM_ID).toString();
            System.out.println("The Item ID of this part is "+itemID);
        }

    }

}
```

7.3.4.1.3 The SaveManagerSession Class

The CATIA information structure is driven by the Integration interface. As mentioned in the Save Manager Handler Definition section, com.ebsolutions.savemanager.SaveManagerSession class is required to create the SaveManagerHandler. The com.ebsolutions.savemanager.SaveManagerSession class is the entry point for the CATIA structure in every display object and action.

As mentioned in the Objects Accessor Methods section, the Application can be accessed from every object using the getApplication() method. The Application contains the SaveManagerSession (the SaveManagerHandler populates the Application with the SaveManagerSession object). Therefore, if you can access the Application, you can access the SaveManagerSession using the getSaveManagerSession() method of the Application class.

7.3.4.2 Concrete Classes Implementation

7.3.4.2.1 Basic Objects Implementations

The Integration provides an implementation for every display class.

If necessary, the implementation and/or abstract classes can be overridden to ensure specific, desired behavior. Overriding concrete classes is helpful for unique needs, such as changing the buttons' display or the toolbar location.

For example:

Override the AbstractSaveModule abstract class rather than the AbstractModule class, or override the SaveModuleToolBarButton class rather than the AbstractDisplayElement, for very specific display and behavior.

The SaveModuleToolBarButton can be overridden to present a default display, such as a JButton.

The AbstractDisplayElement can be customized with a user-specific form, as desired.

Delivery Implementation Classes Available in the Cat2uimanager.jar File

Abstract class	Implementation Classes
AbstractApplication	com.ebsolutions.uimanager.AbstractUIManagerApplication com.ebsolutions.savemanager.SaveManagerApplication
AbstractModule AbstractSaveModule	com.ebsolutions.uimanager.modules.SaveModule
AbstractToolBar	com.ebsolutions.uiapplication.ui.UIApplicationToolBar
AbstractDisplayElement	com.ebsolutions.uimanager.modules.SaveModuleToolBarButton com.ebsolutions.uimanager.modules.SaveModuleToolBarCheckBox com.ebsolutions.uimanager.modules.SaveModuleSwingToolBarButton

7.3.4.2.2 Complex Objects Implementation

The Cat2uimanager.jar file also contains some complex objects implementations, found in the **com.ebsolutions.uimanager.modules** package, and can be customized. For instance, customize the internal display of the Save Module or change the part editor display.

- All classes of the com.ebsolutions.uimanager.modules package are concrete implementations of display objects.
- The entry point to use these implementations is the SaveModule. In other words, a module must be redefined to override the SaveModule and change elements of the SaveModule display.

SaveModule Methods

- `createDefaultTable()`: customizes the table structure representation.
- `createDefaultTableEditor()`: customizes the part editor (right pane of the Save Manager display).
- `createDefaultTableModel()`: customizes CATIA data representation

For example, redefine the `createDefaultTableEditor()` SaveModule method, then return your own implementation of the com.ebsolutions.uimanager.modules.TableEditor.

7.3.4.3 Data and Objects Accessors

Data Accessor Methods
Objects Accessor Methods
User Interface Classes

After objects are created by factories, the SaveManagerHandler calls the **public void initialize(Attributes attributes)** method of the newly created objects. This method is used for the instantiated class when the class is extending the expected Abstract class as described earlier. The Abstract classes are designed to deliver existing useful methods for an easier customization.

The public void initialize(Attributes attributes) method delegates the xml attributes management to the object itself.

For example:

A DisplayElement tag creates an object extending the AbstractDisplayElement class and the AbstractDisplayElement class contains the initialize method.

In this case, the created object manages all the attributes of the DisplayElement tag itself, such as storing the icon path, the corresponding action, etc.

If attributes are required in the *savemanager.xml* file for a specific tag, refer to the following code to override the public void initialize(Attributes attribute) method in a redefined concrete class.

```
  public ConcreteClass extends AbstractModule
                           |
                           |
                           |
  public void initialize(Attributes attributes)
  {
         super.initialize(attributes);
         String newAttributeName =
attributes.getValue("newAttribute");
  }
```

The `initialize` method is the final xml data population step. After this function, objects corresponding to the *savemanager.xml* definition are created and populated with all necessary xml values. The next step is to create the user interface classes.

7.3.4.3.1 Data Accessor Methods

Every main class represents a display element. The AbstractApplication class (the starting point for the display) extends the JFrame java class. An *application redefinition* results in changes to the Save Manager frame. Every main class, except the AbstractApplication, does not extend any Java component directly. The classes use an abstract **createComponent** method which must be defined in a concrete class in order to return a JComponent.

Abstract classes store the standard xml data describing the element. Additonal xml data can be added and stored in the concrete class by overriding the initialize(Attributes) method. By default, the method to access the data is as follows. The three basic methods depend on the abstract class, however, one retrieve method exists for each xml data corresponding to your class.

- `getIcon()` : returns the image attribute value present in the xml
- `getName()` : returns the identifier attribute value present in the xml
- `getNameRef()` : returns the nameRef attribute value present in the xml

If custom xml attributes are added, override the initialize(Attributes) method and the create the accessor method. Refer to the table below, if necessary.

Basic Data Accessors For Different Objects

XML Tags	Main Class	Attribute	Method Accessor	Comments
<UIApplication>	AbstractApplication	nameRef	getNameRef()	---
<UIApplication>	AbstractApplication	image	getApplicationIcon()	Returns an Image object

XML Tags	Main Class	Attribute	Method Accessor	Comments
<UIApplication>	AbstractApplication	sizex,sizey	getApplicationSize()	Returns a Point object
<UIApplication>	AbstractApplication	positionx,positiony	getApplicationPosition()	Returns a Point object
<SaveManagerToolBar,Toolbar>	AbstractToolbar	nameRef	getNameRef()	---
<SaveManagerToolBar,Toolbar>	AbstractToolbar	name	getName()	---
<Module>	AbstractModule	name	getName()	---
<Module>	AbstractModule	nameRef	getNameRef()	
<Module>	AbstractModule	name	getIcon()	Returns an Image object
<DisplayElement>	AbstractDisplayElement	identifier	getName()	---
<DisplayElement>	AbstractDisplayElement	nameRef	getNameRef()	---
<DisplayElement>	AbstractDisplayElement	action	getAction()	Returns an AbstractAction object

XML Tags	Main Class	Attribute	Method Accessor	Comments
<DisplayElement>	AbstractDisplayElement	image	getIcon()	Returns an Image object
<DisplayElement>	AbstractDisplayElement	showTitle	isTitleShow()	Returns a boolean

7.3.4.3.2 Objects Accessor Methods

The Save Manager process automatically creates a hierarchy between classes.

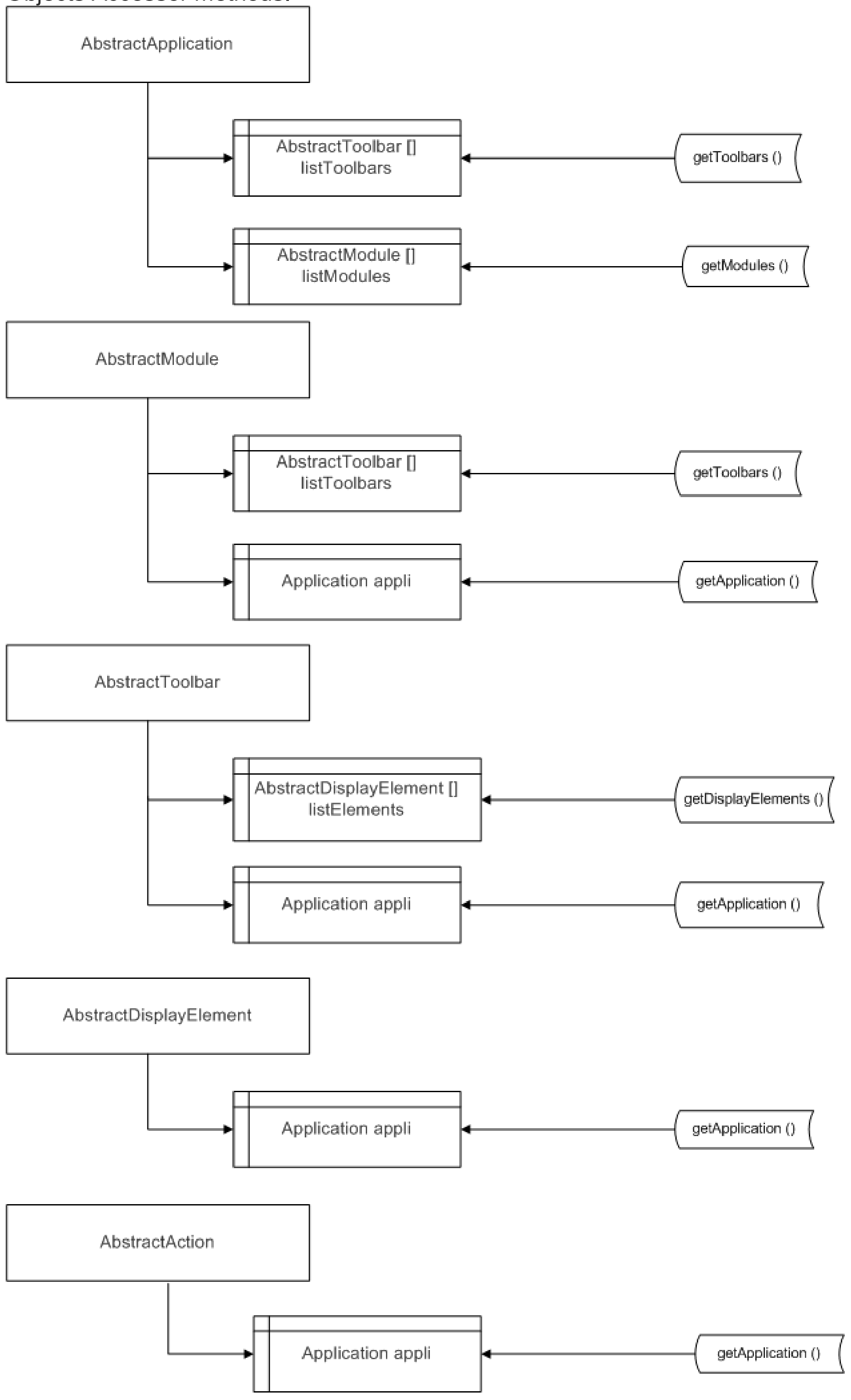
For example, if an application is defined in the xml file using the <UIApplication> tag and two toolbars are defined within this tag, the toolbars can be accessed in the application redefinition using the getToolbars() method, which returns the AbstractToolbar tab. Then, place the toolbars into the concrete class defined in the xml file to get the full concrete class methods accessible.

Refer to the following list for a list of the Objects Accessor Methods.

- `getToolBars()` : returns the toolbar list of the object. Method presents in the AbstractModule and AbstractApplication classes.
- `getModules()` : returns the modules list of the object. Method presents in the AbstractApplication class.
- `getDisplayElements()` : returns the display element list of the object. Method presents in the AbstractToolBar class.
- `getApplication()` : returns the application object. Method presents in every Abstract class. The getApplication() method exists in every class which is helpful when retrieving the first level object (the application). From the first level, every other display object and a CATIA session can be accessed. Every object created by the factories contains its Application information (the application is populated in the newly created object by the SaveManagerHandler).

The following schema summarizes the Objects Accessor Methods. Note that the AbstractSaveModule class extends the AbstractModule class, and has, by extension, the same

Objects Accessor methods.



7.3.4.3.3 User Interface Classes

Several abstract classes are used to create the Save Manager display. These abstract classes provide and suggest methods to create java graphical objects. The list below contains available abstract classes known as **main classes** because of their lowest level within the Save Manager hierarchy.

- `AbstractApplication` : when extended, creates a new Save Manager Application.
- `AbstractModule` : when extended, creates a new Module.
- `AbstractSaveModule` : when extended, creates a new Module with the SaveModule capabilities.
- `AbstractTableEditor` : when extended, changes the SaveModule table editor.
- `AbstractToolBar` : when extended, creates a new toolbar.
- `AbstractDisplayElement` : when extended, creates a new toolbar element.

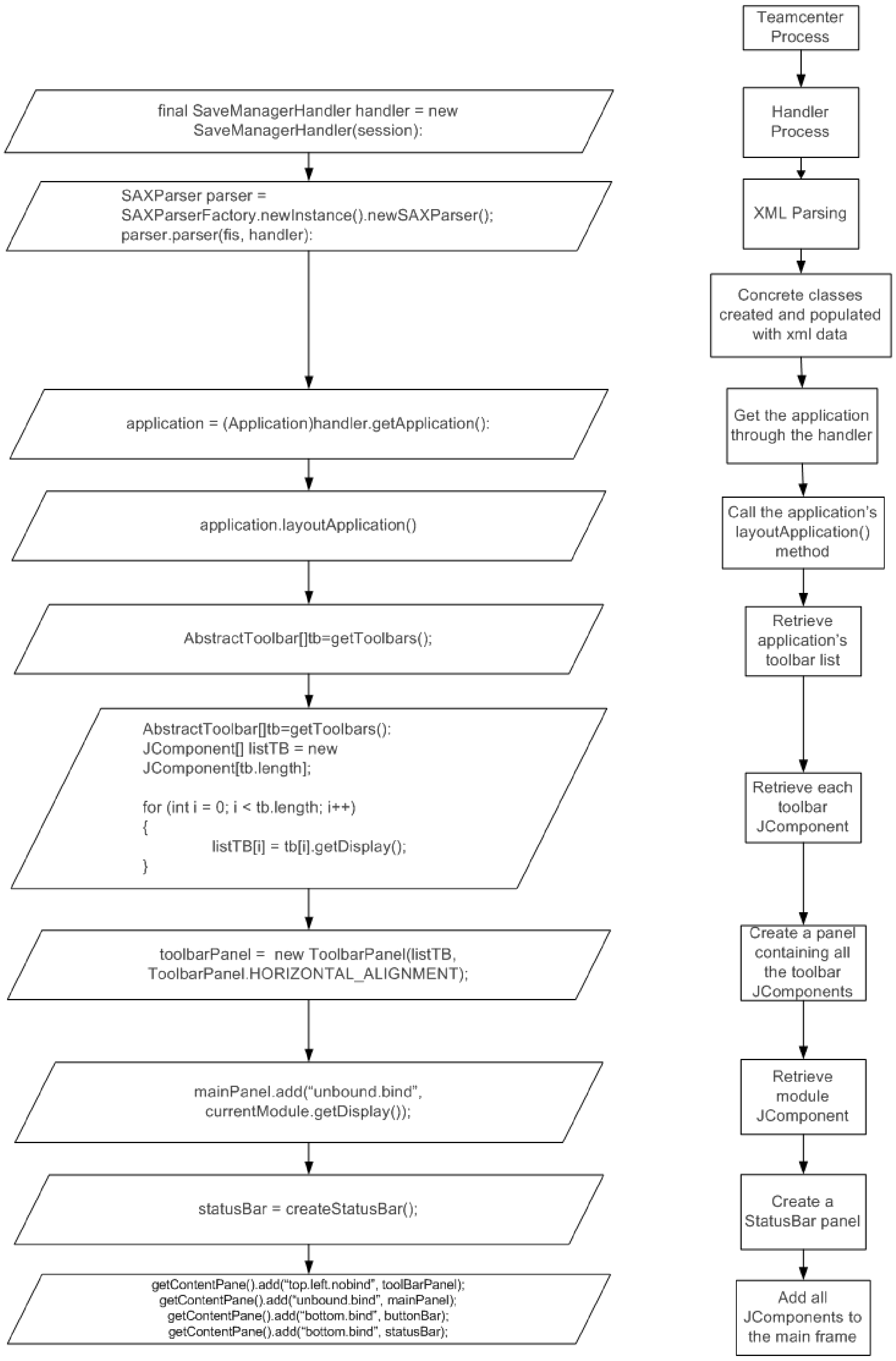
The `AbstractTableEditor` class cannot be customized inside the savemanager.xml file. Create a class extending the SaveModule then redefine a new TableEditor to create a new Table Editor.

7.3.4.4 Interface Construction

The Objects Construction Hierarchy
Display Methods

7.3.4.4.1 The Objects Construction Hierarchy

At this stage of development, every object is ready to be displayed. The main display object is the Application because it is the JFrame that includes all Save Manager display objects. Review the schema below describing the lifecycle of the Save Manager Frame construction. Note that concrete classes are identified in the xml file by the `class` attribute value. Concrete classes extend the many abstract classes described throughout this document.



```
final SaveManagerHandler handler = new
SaveManagerHandler(session):

SAXParser parser =
SAXParserFactory.newInstance().newSAXParser();
parser.parser(fis, handler):

application = (Application)handler.getApplication():

application.layoutApplication()

AbstractToolbar[]tb=getToolbars();

AbstractToolbar[]tb=getToolbars():
JComponent[] listTB = new
JComponent[tb.length];

for (int i = 0; i < tb.length; i++)
{
            listTB[i] = tb[i].getDisplay();
}

toolbarPanel =  new ToolbarPanel(listTB,
ToolbarPanel.HORIZONTAL_ALIGNMENT);

mainPanel.add("unbound.bind",
currentModule.getDisplay());

statusBar = createStatusBar();

getContentPane().add("top.left.nobind", toolBarPanel);
getContentPane().add("unbound.bind", mainPanel);
getContentPane().add("bottom.bind", buttonBar);
getContentPane().add("bottom.bind", statusBar);
```

Teamcenter Process
Handler Process
XML Parsing
Concrete classes created and populated with xml data
Get the application through the handler
Call the application's layoutApplication() method
Retrieve application's toolbar list
Retrieve each toolbar JComponent
Create a panel containing all the toolbar JComponents
Retrieve module JComponent
Create a StatusBar panel
Add all JComponents to the main frame

The **layoutApplication** method is provided by the SaveManagerApplication class and is an implementation of the AbstractUIManagerApplication.

The **AbstractUIManagerApplication** class does not provide a method to display it. To implement a new Application:

1. Extend the com.ebsolutions.uimanager.AbstractUIManagerApplication class. This extends the SaveManagerApplication which extends the AbstractUIManagerApplication class.

2. Redefine the layoutApplication method to modify the component display.

The new class must be an extension of com.ebsolutions.uimanager.AbstractUIManagerApplication class because the Integration only handles this class type.

7.3.4.4.2 Display Methods

In the schema above, different objects are responsible for their display. Each object displays itself by returning a JComponent corresponding to it. Customize display objects by redefining their methods.

Class	Display Method
AbstractUIManagerApplication	No method. Redefines the layoutApplication method in an implementation of the com.ebsolutions.uimanager.AbstractUIManagerApplication class
AbstractToolBar	getDisplay()
AbstractModule	getDisplay()
AbstractDisplayElement	getDisplay()

Every object can retrieve objects composing its display, as described in the Objects Accessor Methods section. For example, the application is able to retrieve toolbars with the getToolbars() method.

Redefine the **getDisplay()** method of the parent display to accommodate the children display. All getDisplay() methods are implemented in the Save Manager the same way. For example, create a new module implementation of the com.ebsolutions.savemanager.modules.SaveModule class, then redefine the getDisplay() method to return a JComponent containing the toolbar on the bottom of the display which will display the toolbar on the bottom of the Save Manager.

When implementing such a class, ensure the createComponent() method is redefined and not the getDisplay() method because the getDisplay()method calls the createComponent() method of the object, by default.

7.3.4.5 Managing XML Information

7.3.4.5.1 The SAX Parser

Simple API for XMLSAX, also known as **SAX**, was originally a Java-only API. SAX was the first widely adopted API for XML in Java. However, there are versions for several programming language environments other than Java. This API is included with Java and is used in the Save Manager for decoding the *savemanager.xml* file.

The SAX Parser instance can be retrieved as follows :

```
SAXParser parser =
SAXParserFactory.newInstance().newSAXP
arser();
```

To initiate the parse, an input stream and handler must be provided:

```
parser.parse(fis, handler);
```

For the Integration, the input stream is the file *savemanager.xml* and the Handler is the *SaveManagerHandler*.

7.3.4.5.2 *Save Manager Handler Definition*

The **com.ebsolutions.uiapplication.handlers.SaveManagerHandler** class is the main class for the xml data management. It is responsible for the xml information propagation and the first step for the graphical objects creation. The code below initiates a handler to read the savemanager.xml file.

The SaveManagerHandler class is located in the *Cat2uimanager.jar* file and extends the SAX class **org.xml.sax.helpers.DefaultHandler**. After this code, when the parse method is called, the SaveManagerHandler class automatically receives the described data provided by the xml stream.

```
final String catCode = System.getProperty(TCICV5_DIR);


final FileInputStream fis = new FileInputStream(catCode + "/env/" +
SAVEMANAGER_FILE);


final SaveManagerSession session = new SaveManagerSession();

session.setCatiaSession(catSession);



// parsing savemanager.xml.


final SaveManagerHandler handler = new SaveManagerHandler(session);
try {
    SAXParser parser =
SAXParserFactory.newInstance().newSAXParser();
    parser.parse(fis, handler); // START THE XML PARSE
    parser = null;
    } catch (IOException ex)
    {
   ex.printStackTrace();
    }
    catch (ParserConfigurationException ex)
    {
    ex.printStackTrace();
    }
    catch (SAXException ex)
    {
    ex.printStackTrace();
    }
```

7.3.4.5.3 *Redefine the Number of DefaultHandler Methods*

The com.ebsolutions.uiapplication.handlers.SaveManagerHandler class extends the SAX class org.xml.sax.helpers.DefaultHandler, required for the SAX logic. This class redefines the number of the DefaultHandler methods in order to catch the events of the SAX handler.

Two common methods within the SaveManagerHandler class are:

- `public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException`
- `public void endElement(String uri, String localName, String qName) throws SAXException`

7.3.4.5.4 *Definitions*

The **startElement** function is called each time a tag and its attributes are decoded from the xml stream.

The **endElement** function is called each time a close tag is encountered in the xml stream, indicating the tag definition is ending.

The **qName** string determines the kind of object described in the xml file and its properties. The qName contains the name of the tag currently decoded. For instance:

- <UIApplication>
- <SaveManagerToolBar>
- <Module>
- <DisplayElement>

The `attributes` attribute argument contains every attribute definition of the qName Tag.

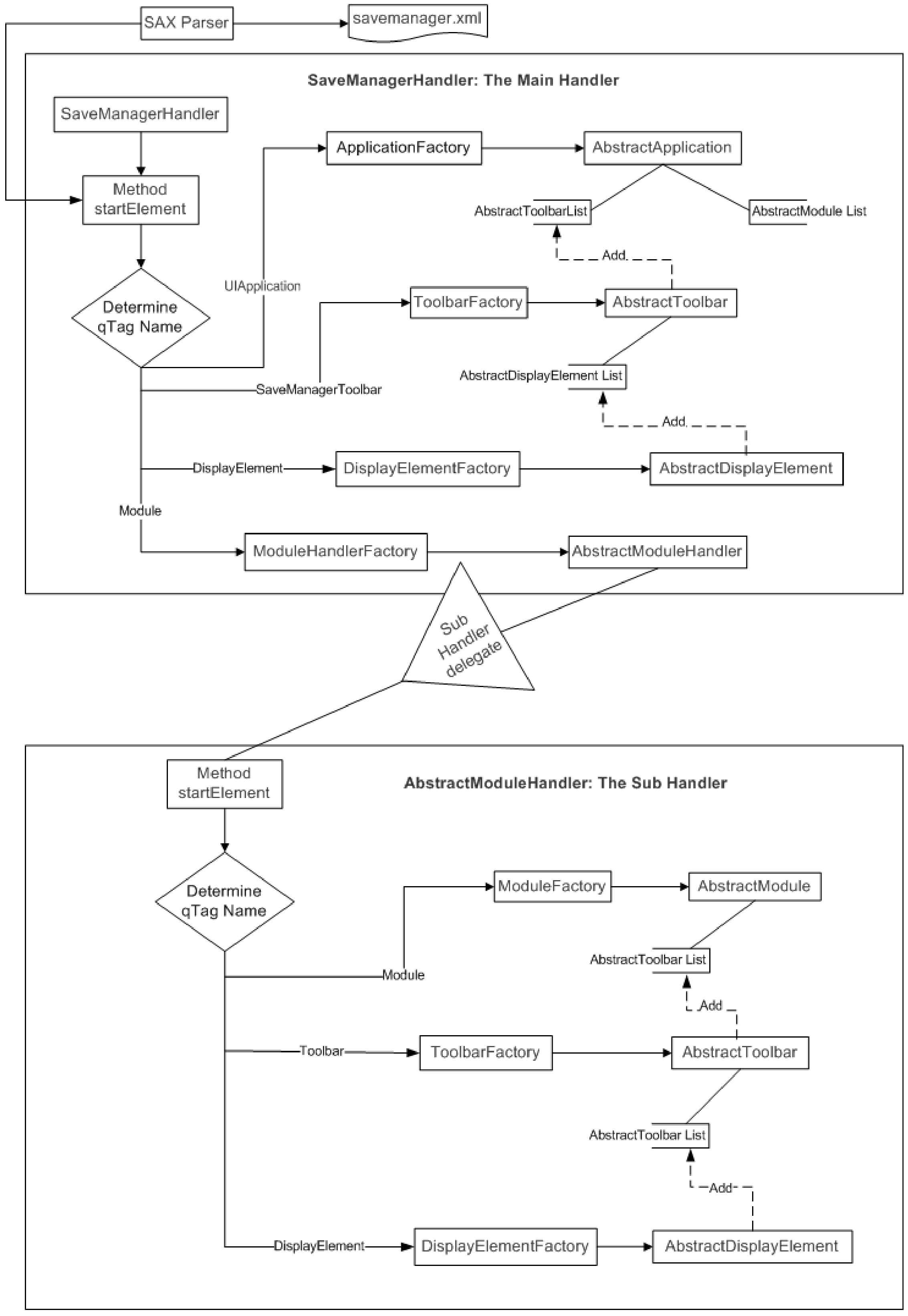
The `class` attribute indicates which class will be instantiated. The SaveManagerHandler uses this attribute to create the corresponding Factories Mechanism.

7.3.4.5.5 Save Manager Handler Tasks

The SaveManagerHandler:

- retrieves tag information from the xml stream
- creates the classes corresponding to the xml definitions
- initializes the objects with the attributes values provided by the xml
- delegates the handler values into the Sub Handler.

The following schema describes the Factory Mechanism used to create the objects corresponding to the xml tags. The **Method startElement** of the handler analyzes the tags, retrieves the `class` attribute of each tag and uses the Factory Mechanism to instantiate the class corresponding to the `class` attribute value.


SAX Parser
savemanager.xml
SaveManagerHandler: The Main Handler
SaveManagerHandler
Method startElement
Determine qTag Name
UIApplication
ApplicationFactory
AbstractApplication
AbstractToolbarList
AbstractModule List
Add.
ToolbarFactory
AbstractToolbar
SaveManagerToolbar
AbstractDisplayElement List
Add.
DisplayElement
DisplayElementFactory
AbstractDisplayElement
Module
ModuleHandlerFactory
AbstractModuleHandler
Sub Handler delegate
Method startElement
AbstractModuleHandler: The Sub Handler
Determine qTag Name
ModuleFactory
AbstractModule
Module
AbstractToolbar List
Add
Toolbar
ToolbarFactory
AbstractToolbar
AbstractToolbar List
Add
DisplayElement
DisplayElementFactory
AbstractDisplayElement

This mechanism results in a list of objects corresponding to the xml definitions. The SaveManagerHandler automatically populates the created objects with their necessary objects. For example, at the end of the SaveManageToolBar definition, the created toolbar is added to the toolbar list in the application. Notice the SaveManagerHandler delegates handling events to an **AbstractModuleHandler**. This process permits customization of the xml file, so that it is possible to implement customized tags within the savemanager.xml.

The **Sub Handler** is only available inside the `Module` tag. The `Module` tag of the *savemanager.xml* file contains the attribute `moduleHandler`, permitting customization of the Module Handler. For example, a CustomModuleHandler class can be created to extend the the AbstractModuleHandler class by replacing the `moduleHandler` attribute value with the path to your CustomModuleHandler. In this case, every tag inside your `Module` definition is received. To retrieve the xml tags inside the `Module` tag, redefine the startElement and endElement functions inside the CustomModuleHandler class.

An `action` attribute is defined on the `DisplayElement` tag, and is managed by the AbstractDisplayElement itself. After the element is created by the corresponding factories, an initialization method is called and the attributes list of the tag is passed into this function to determine attribute management. In the case of the `DisplayElement` object, the initialization method retrieves the `action` attribute value and calls the ActionFactory to instantiate the corresponding action.

7.3.4.5.6 Custom Handler Examples

Redefine the Handler with Additional Tags

The following example demonstrates placing a new xml tag with the custom handler by redefining the handler with additional tags. In this example, redefining the handler is performed without redefining the module.

Both the module handler and the module should be redefined because new xml tag values must be retrieved. A later example addresses developing a complete module and handler. The purpose of this example is only to create and manage xml tags.

Refer to the following values used to add the new tag (`Fruit`) to the *savemanager.xml* file and replace the `moduleHandler` attribute value with the path to the custom handler.

1. The savemanager.xml definition:

```
<Module nameRef="Save Module"
      name="save_module"
      moduleHandler="com.customcompany.savemanager.handlers.CustomM
oduleHandler"
      image="/com/ebsolutions/savemanager/modules/images/save.png"
      class="com.ebsolutions.savemanager.modules.SaveModule">
      <Fruit name="Apple"/>
      <Fruit name="Lemon"/>
      <Fruit name="Orange"/>
```

2. The `moduleHandler` attribute value.

```
package com.customcompany.savemanager.handlers;
public class CustomModuleHandler extends AbstractModuleHandler
      {
        protected List listFruit;
        public CustomModuleHandler()
                {
                        listFruit  = new Vector();
                }
                public void startElement(String uri, String
localName, String qName,
                                          Attributes attributes)
throws SAXException
               {
                       super.startElement(uri, localName, qName,
attributes);

                       if (qName.equals("Fruit"))
                       {
                               String nameFruit=
attributes.getValue("name");
                               listFruit.add(nameFruit);
                       }
               }
               public void endElement(String uri, String localName,
String qName)
                            throws SAXException
               {
                        super.endElement(uri, localName, qName); }
       }
```

The handler is now able to retrieve the `Fruit` list. When the *savemanager.xml* file parsing is complete, the CustomHandler contains the `Fruit` list in the `listFruit` vector.

Button

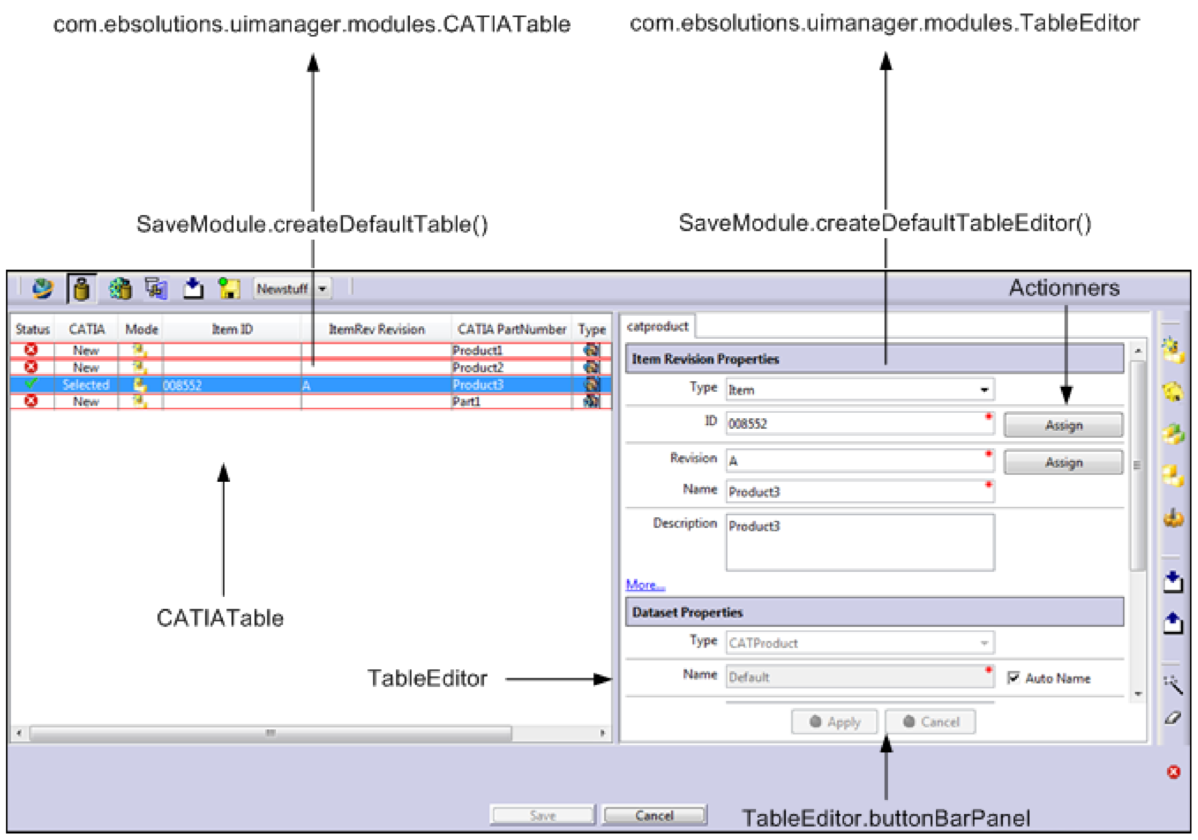
The previous example illustrates adding xml tags in a module. Tags cannot be added on other xml objects. However, other objects can add attributes and be retrieved in the implemented class.

The next example redefines the **initialize(Attribute Method)** to get the color attribute from the xml and apply it to the button in the **createComponent()** method redefinition.

```
public class TestButton extends AbstractDisplayElement
 {
    JButton button = null;
    Color color = null;

    public JComponent createComponent()
    {
       JButton myButton = new JButton();
       button = myButton;
       button.setBorder(null);
       button.setBackground(color);
       button.setText(name);
       button.setIcon(new ImageIcon(getIcon()));
       return myButton;
    }
    public void initialize(Attributes attributes)
    {
       super.initialize(attributes);
       setColor(attributes.getValue("color"));
    }
    protected void setColor(String col)
    {
        try
        {
           color = new Color(Integer.parseInt(col));
        }
        catch (Exception k)
        {
        }
     }
     public String getID()
     {
         return name;
     }
 }
```

7.3.4.6 Save Module Elements

Refer to the descriptions of classes, delivered with the Integration, which compose the Save Module.


com.ebsolutions.uimanager.modules.CATIATable
com.ebsolutions.uimanager.modules.TableEditor
SaveModule.createDefaultTable()
SaveModule.createDefaultTableEditor()
Actionners
Newstuff
CATIATable
TableEditor
TableEditor.buttonBarPanel
Save
Cancel
Apply
Cancel
Assign
Assign
More...

Status	CATIA	Mode	Item ID	ItemRev Revision	CATIA PartNumber	Type
✗	New				Product1	
✗	New				Product2	
✓	Selected		008552	A	Product3	
✗	New				Part1	

catproduct

Item Revision Properties

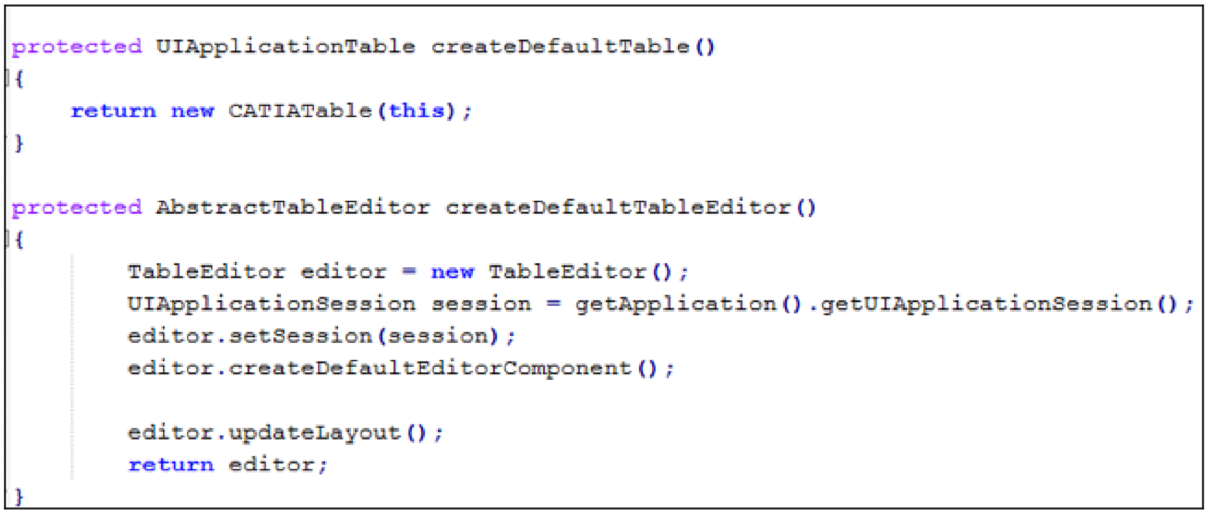
Type: Item
ID: 008552
Revision: A
Name: Product3
Description: Product3

Dataset Properties

Type: CATProduct
Name: Default
☑ Auto Name

Ensure customizations take place on the correct class. Modifying an existing module is easier than redefining a complete module. However, refer to the illustration below to create a model that extends the SaveModule.

Customize the Table and appearance of the Table Editor by redefining the createDefaultTable() and createDefaultTableEditor() methods in an extended class of the SaveModule. Refer to the code below for an example.



```
protected UIApplicationTable createDefaultTable()
{
    return new CATIATable(this);
}

protected AbstractTableEditor createDefaultTableEditor()
{
        TableEditor editor = new TableEditor();
        UIApplicationSession session = getApplication().getUIApplicationSession();
        editor.setSession(session);
        editor.createDefaultEditorComponent();

        editor.updateLayout();
        return editor;
}
```

7.3.4.7 Factories Mechanism

Factories are based on the **java reflection** feature. The goal of a factory is to instantiate a class from its signature. The signature class is the package name and includes the class and the name of the class itself. For example, the signature of the famous JFrame class is *javax.swing.JFrame*.

The factory process is very useful for portable projects. During the Save Manager process, the SaveManagerHandler catches the xml information and uses the factory system to instantiate the classes defined in the `class` attribute of every tag. Because of this, every Save Manager object is independent and each one customized.

The `class` attribute value must be a class extends the corresponding Abstract class. Refer to the following table for the list of the factories and Abstract classes.

7.3.4.7.1 Available Factories

The basic Save Manager package suggests factories for the Main classes of the Save Manager. Refer to the table below for a factory list and the objects managed.

XML Tags	XML Attribute	Factory Class Name	Instantiable Classes

XML Tags	XML Attribute	Factory Class Name	Instantiable Classes
<UIApplication>	class	ApplicationFactory	AbstractApplication
<SaveManagerToolBar>,<ToolBar>	class	ToolBarFactory	AbstractToolBar
<Module>	moduleHandler	ModuleHandlerFactory	AbstractModuleHandler
<Module>	class	ModuleFactory	AbstractModule
<DisplayElement>	class	DisplayElementFactory	AbstractDisplayElement
<DisplayElement>	action	ActionFactory	AbstractAction

The Abstract classes listed are **seed classes**. When a definition is redefined, the class that is redefined must extend the corresponding Abstract class. Abstract classes can be modified to fit classes, as desired.

The Factories classes are not designed to be used independently because the Save Manager uses them to instantiate the classes in the xml file. Ensure the redefined class extends the correct Abstract class corresponding to the schema above. If necessary, refer to the "Custom Handler Example" which illustrates how the CustomModuleHandler extends the abstract class AbstractModuleHandler.

7.3.4.8 The Listeners

Adding Listeners to Specific Implementations

Save Module Listening Capabilities

Listeners, provided in the *out of the box* delivery of the Save Manager, notify different objects of different events.

7.3.4.8.1 Adding Listeners to Specific Implementations

Refer to the schema before adding components as listeners of AbstractUIManagerSession, Application, ListSelection and Reorder objects.

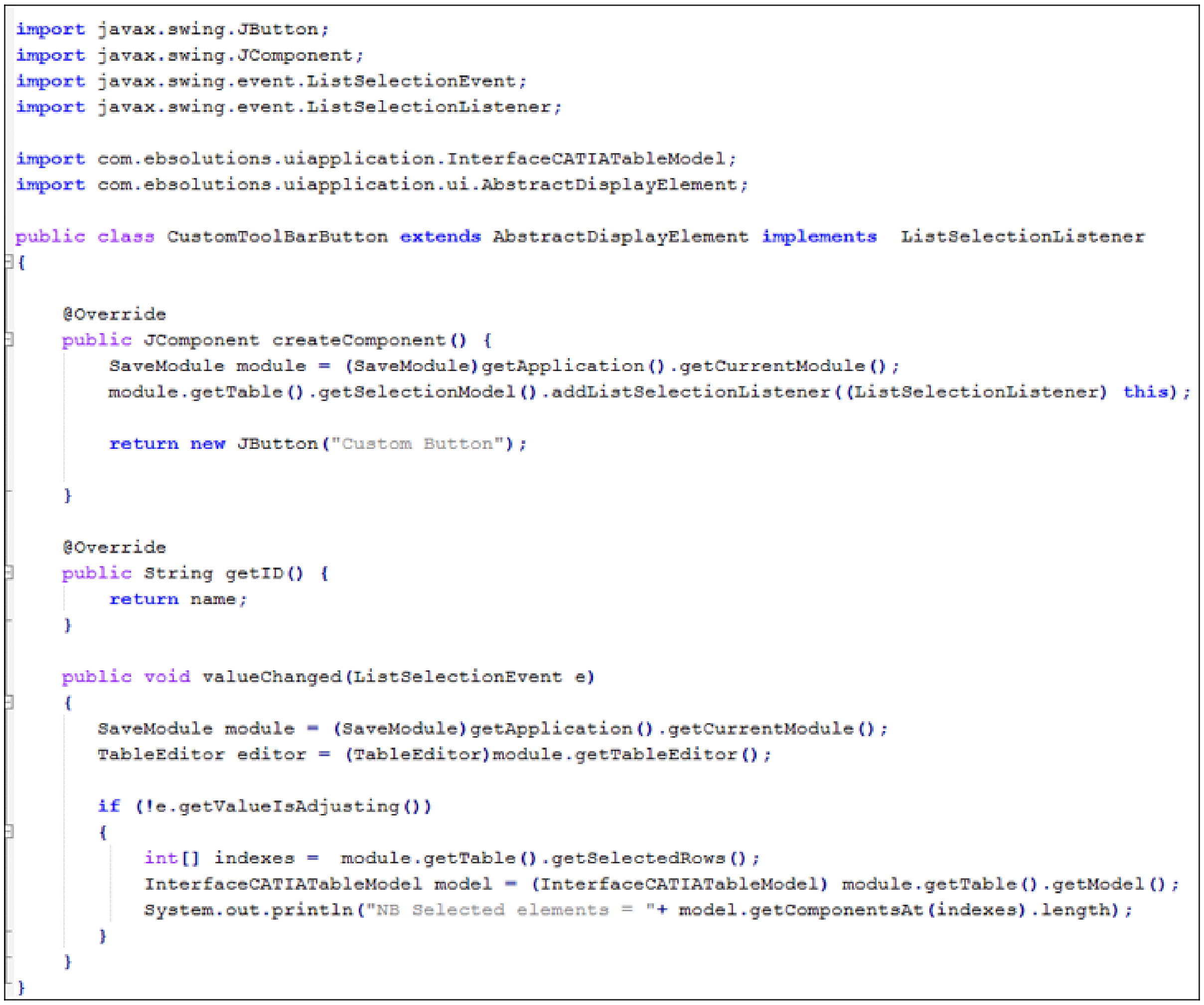
Recall that the AbstractUIManagerSession and the Application classes can be accessed using the getApplication() and getUIApplicationSession().getCatiaSession() methods. **ListSelection** and **ReorderModel** are also accessible by casting the module into a SaveModule, retrieving the table and its models, then adding your component as a listener to them. This will cause notifications each time the event is initiated.

Display Element Implementation

Refer to the sample code below to create a display element implementation to notify each time a selection changes in the Save Table. Elements selected can also be accessible using the `tableModel getComponentsAT` method.

An overview of the steps is as follows:

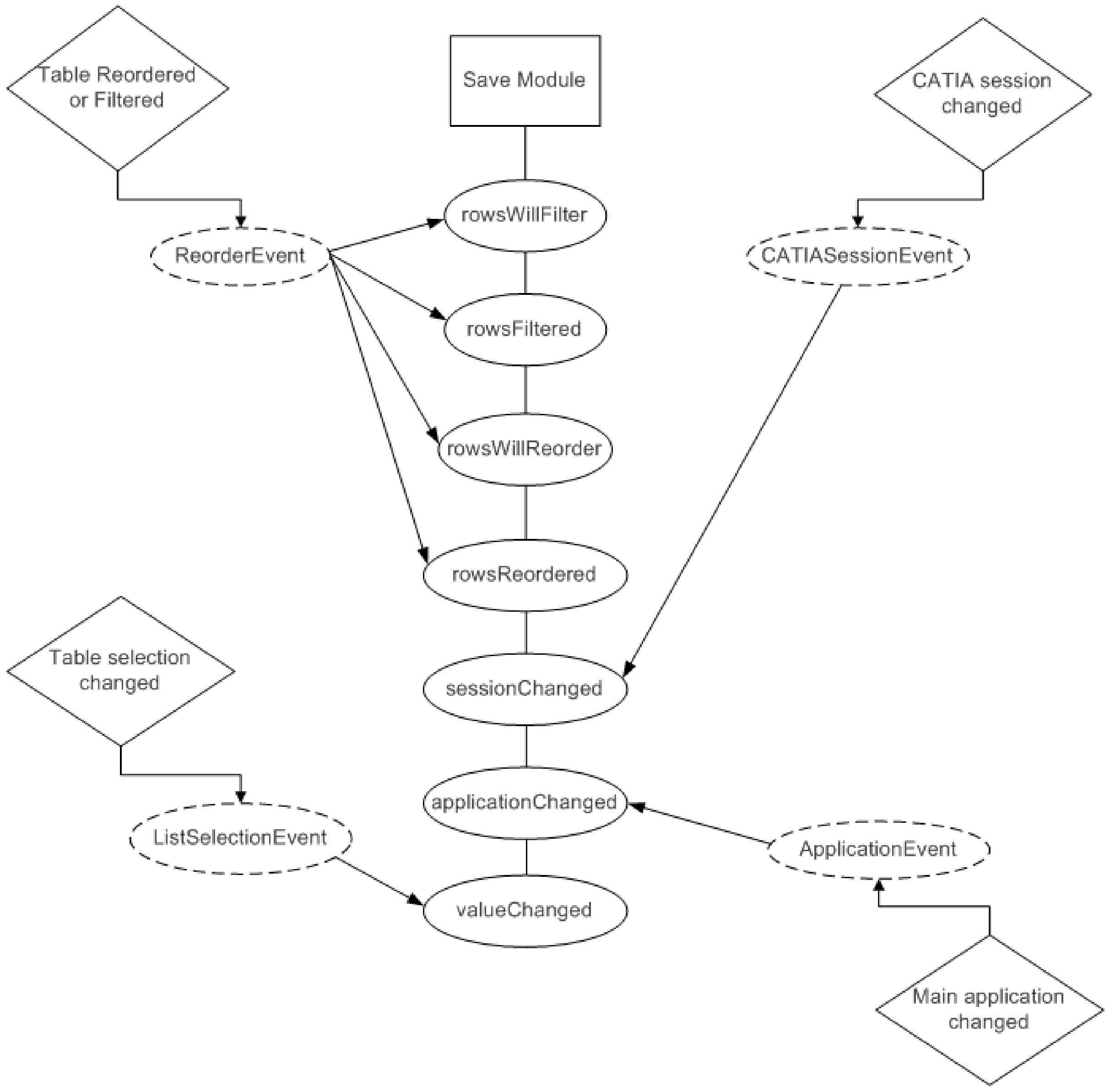
1. Implement the `ListselectionListener` interface.
2. Add your component as a listener to the table selection model.
3. Redefine the `valueChanged(ListSelectionEvent e)` method to get the new selected components.



```java
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import com.ebsolutions.uiapplication.InterfaceCATIATableModel;
import com.ebsolutions.uiapplication.ui.AbstractDisplayElement;

public class CustomToolBarButton extends AbstractDisplayElement implements  ListSelectionListener
{

    @Override
    public JComponent createComponent() {
        SaveModule module = (SaveModule)getApplication().getCurrentModule();
        module.getTable().getSelectionModel().addListSelectionListener((ListSelectionListener) this);

        return new JButton("Custom Button");

    }

    @Override
    public String getID() {
        return name;
    }

    public void valueChanged(ListSelectionEvent e)
    {
       SaveModule module = (SaveModule)getApplication().getCurrentModule();
       TableEditor editor = (TableEditor)module.getTableEditor();

       if (!e.getValueIsAdjusting())
       {
           int[] indexes =  module.getTable().getSelectedRows();
           InterfaceCATIATableModel model = (InterfaceCATIATableModel) module.getTable().getModel();
           System.out.println("NB Selected elements = "+ model.getComponentsAt(indexes).length);
       }
    }
}
```

7.3.4.8.2 Save Module Listening Capabilities

When customization of Save Manager elements is required, you will get the events inside the **module**. The module is the main element dispatching events to other objects. The Save Module **listens to** (receives) four main events, as illustrated below.



Event	Description
AbstractUIManagerSessionEvent	Initiated each time an element in the CATIA session changes, such as when a user modifies a property on at least one AbstractUIManagerComponent. The Save Manager display is updated depending on the affected module.

Event	Description
ApplicationEvent	Initiated each time the Save Manager displays. The Save Manager becomes "invisible" when OK is selected during the Save process. When the Save process is initiated again, the ApplicationEvent scans every listener before setting the Save Manager visible again. The Save Module updates the display after receiving this event.
ReorderEvent	Initiated each time a column in the Save Table is sorted or filtered. The ReorderEvent is dispatched in four functions : two reorder operations and two filter operations. The *will* function is initiated before the reorder is performed and the *ed* function is initiated after the reorder is performed. This mechanism keeps the table selection information in the *will* functions and resets the selection in the *ed* functions.
ListSelectionEvent	Initiated each time an element selection changes in the Save Table. This event is pre-implemented by the AbstractSaveModule class. The AbstractSaveModule populates the TableEditor with the selected AbstractUIManagerComponents and displays the correct editor, depending on the selection.

7.3.5 Save Manager Workbench

The Cat2uimanager.jar File

Setting up an Environment

The Save Manager Development Toolkit contains a specific structure. Refer to the sub-topics above to begin a customization project.

7.3.5.1 The Cat2uimanager.jar File

Packages

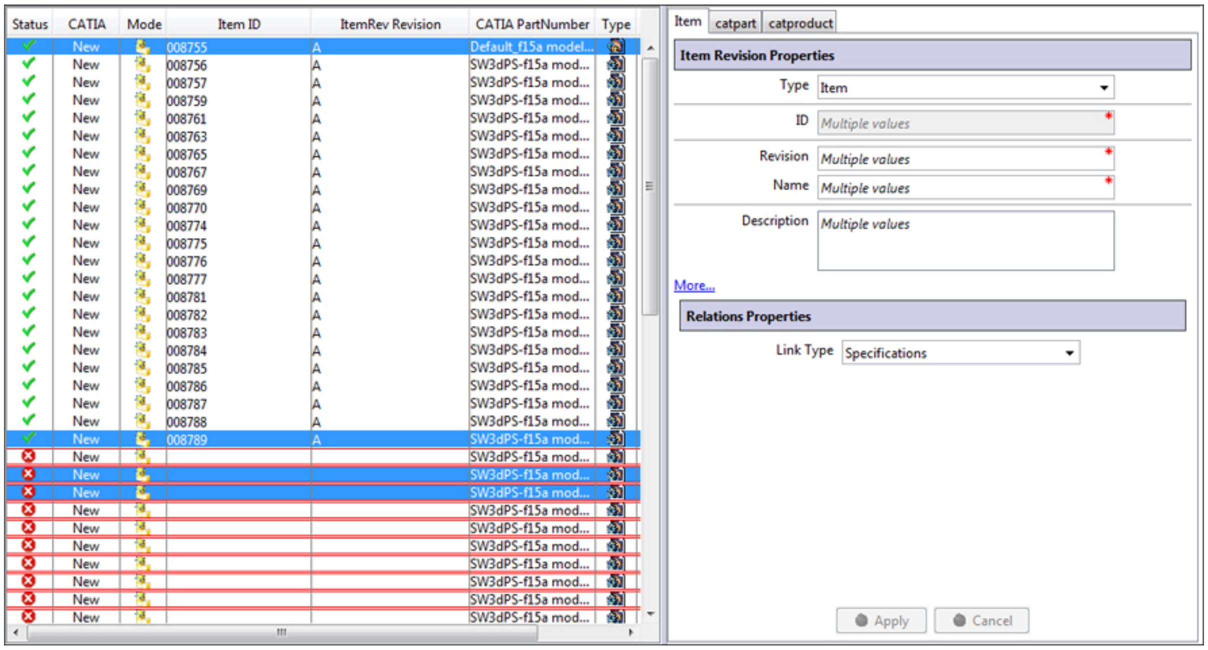
The *client* sub-directory, within the Integration directory, contains the necessary *.jar* files used to create customizations.

The *Cat2uimanager.jar* file is the package distribution that contains all the necessary classes used to implement your own objects. Use the com.ebsolutions.uiapplication package to create a new Save Manager interface, with specific, customized behavior.

The Teamcenter implementation of the Save Manager contains complete data management and data representations are redefined in the jar file (in the InterfaceCATIASession and InterfaceCATIAcomponent interfaces) to fit Teamcenter metadata requirements. The implementation also drives the Save Manager data with the Teamcenter Server to synchronize visuals and metadata.

Use the com.ebsolutions.uimanager package in your project for Teamcenter specific customizations, rather than creating a new implementation. This package contains implementations to manipulate Teamcenter data, and should be used as a source, rather than implementing a complete, new specific implementation.

Refer to the JAVA Code Customizations section for more information related to implementing a JAVA project.



Status	CATIA	Mode	Item ID	ItemRev Revision	CATIA PartNumber	Type
✓	New		008755	A	Default_f15a model...	
✓	New		008756	A	SW3dPS-f15a mod...	
✓	New		008757	A	SW3dPS-f15a mod...	
✓	New		008759	A	SW3dPS-f15a mod...	
✓	New		008761	A	SW3dPS-f15a mod...	
✓	New		008763	A	SW3dPS-f15a mod...	
✓	New		008765	A	SW3dPS-f15a mod...	
✓	New		008767	A	SW3dPS-f15a mod...	
✓	New		008769	A	SW3dPS-f15a mod...	
✓	New		008770	A	SW3dPS-f15a mod...	
✓	New		008774	A	SW3dPS-f15a mod...	
✓	New		008775	A	SW3dPS-f15a mod...	
✓	New		008776	A	SW3dPS-f15a mod...	
✓	New		008777	A	SW3dPS-f15a mod...	
✓	New		008781	A	SW3dPS-f15a mod...	
✓	New		008782	A	SW3dPS-f15a mod...	
✓	New		008783	A	SW3dPS-f15a mod...	
✓	New		008784	A	SW3dPS-f15a mod...	
✓	New		008785	A	SW3dPS-f15a mod...	
✓	New		008786	A	SW3dPS-f15a mod...	
✓	New		008787	A	SW3dPS-f15a mod...	
✓	New		008788	A	SW3dPS-f15a mod...	
✓	New		008789	A	SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	
✗	New				SW3dPS-f15a mod...	

Item Revision Properties

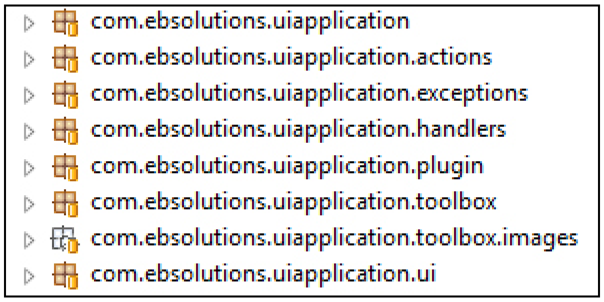
Type	Item
ID	Multiple values
Revision	Multiple values
Name	Multiple values
Description	Multiple values

Relations Properties

Link Type	Specifications

7.3.5.1.1 *Packages*

Refer to the illustration and the table below for the available Cat2uimanager.jar file packages:

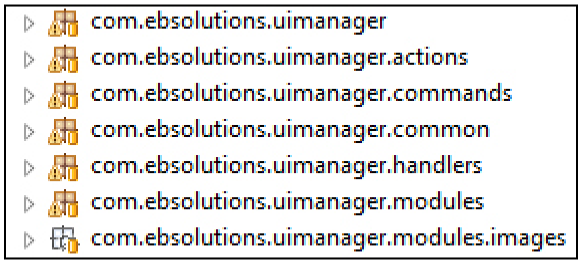


com.ebsolutions.uiapplication
com.ebsolutions.uiapplication.actions
com.ebsolutions.uiapplication.exceptions
com.ebsolutions.uiapplication.handlers
com.ebsolutions.uiapplication.plugin
com.ebsolutions.uiapplication.toolbox
com.ebsolutions.uiapplication.toolbox.images
com.ebsolutions.uiapplication.ui

Package	Contents
com.ebsolutions.uiapplication	Contains the basic interfaces available to implement. It is not necessary to use these classes unless redefining the interfaceAbstractUIManagerSession and InterfaceCATIAComponent interfaces is desired.

Package	Contents
com.ebsolutions.uiapplication.actions	Contains the AbstractAction class. This class can be extended, for example, for a specific button action customization.
com.ebsolutions.uiapplication.exceptions	Contains the SaveManagerException class and is empty. This class is reserved for future use.
com.ebsolutions.uiapplication.handlers	Contains all the Save Manager handler specific classes. It is not necessary to override these classes (these are the Save Manager engine).
com.ebsolutions.uiapplication.toolbox	Contains a set of re-usable graphic elements which can be used without any other package restraints. This package includes elements such as CATIA buttons and toolbar styles. Additional classes included are the FilterModel (for a table) and the OrderedMap class, which provides a map to control the order of populated elements. This package can be used outside the Save Manager Toolkit for another java project.
com.ebsolutions.uiapplication.ui	Contains all the abstract classes to extend when implementing a new graphical object. Classes in this package include the AbstractApplication, AbstractModule, AbstractSaveModule, AbstractToolBar and AbstractDisplayElement classes, and provides the basic behaviors for objects, as described within the Save Manager LifeCycle section.

VerticalLayout and HorizontalLayout classes can be helpful for a new Application implementation. These layouts are complete and display graphic elements easily.



```
▷ com.ebsolutions.uimanager
▷ com.ebsolutions.uimanager.actions
▷ com.ebsolutions.uimanager.commands
▷ com.ebsolutions.uimanager.common
▷ com.ebsolutions.uimanager.handlers
▷ com.ebsolutions.uimanager.modules
▷ com.ebsolutions.uimanager.modules.images
```

Package	Contents
com.ebsolutions.uimanager	Contains the implementations for the corresponding interfaces including AbstractUIManagerSession and AbstractUIManagerComponent Application classes (providing behavior to the Teamcenter version of the Save Manager).
com.ebsolutions.uimanager.actions	Contains all Teamcenter-specific actions. The actions behind the Save Manager buttons are listed within this package and redefined within the savemanager.xml file. Extend one of those to redefine the corresponding action as necessary.
com.ebsolutions.uimanager.commands	Contains specific commands, such as Check-in and Check-out, and must be accessible from CATIA. This package is not re-usable because these commands cannot be redefined through the savemanager.xml file.
com.ebsolutions.uimanager.handlers	Contains the SaveModuleHandler class used to manage the xml tags for the SaveModule. This sub-handler redefines tages such as the `Column` tag used in the Save Table. A specific Handler redefinition can exist to extend it in order to add new xml tag management.

Package	Contents
com.ebsolutions.uimanager.modules	Contains the SaveModule definition and other classes specific to the Teamcenter SaveManager implementation, such as the TableEditor, TableModel and Actionners. Use this package when overriding specific SaveModule parts.

7.3.5.2 Setting up an Environment

A Teamcenter plug-in, such as Eclipse, must be used to customize Teamcenter.

Several IDEs are available to set up a Save Manager project. The following information explains how to set up an Eclipse project. Eclipse is free, user-friendly and very easy to use.

7.3.5.2.1 Configure Eclipse

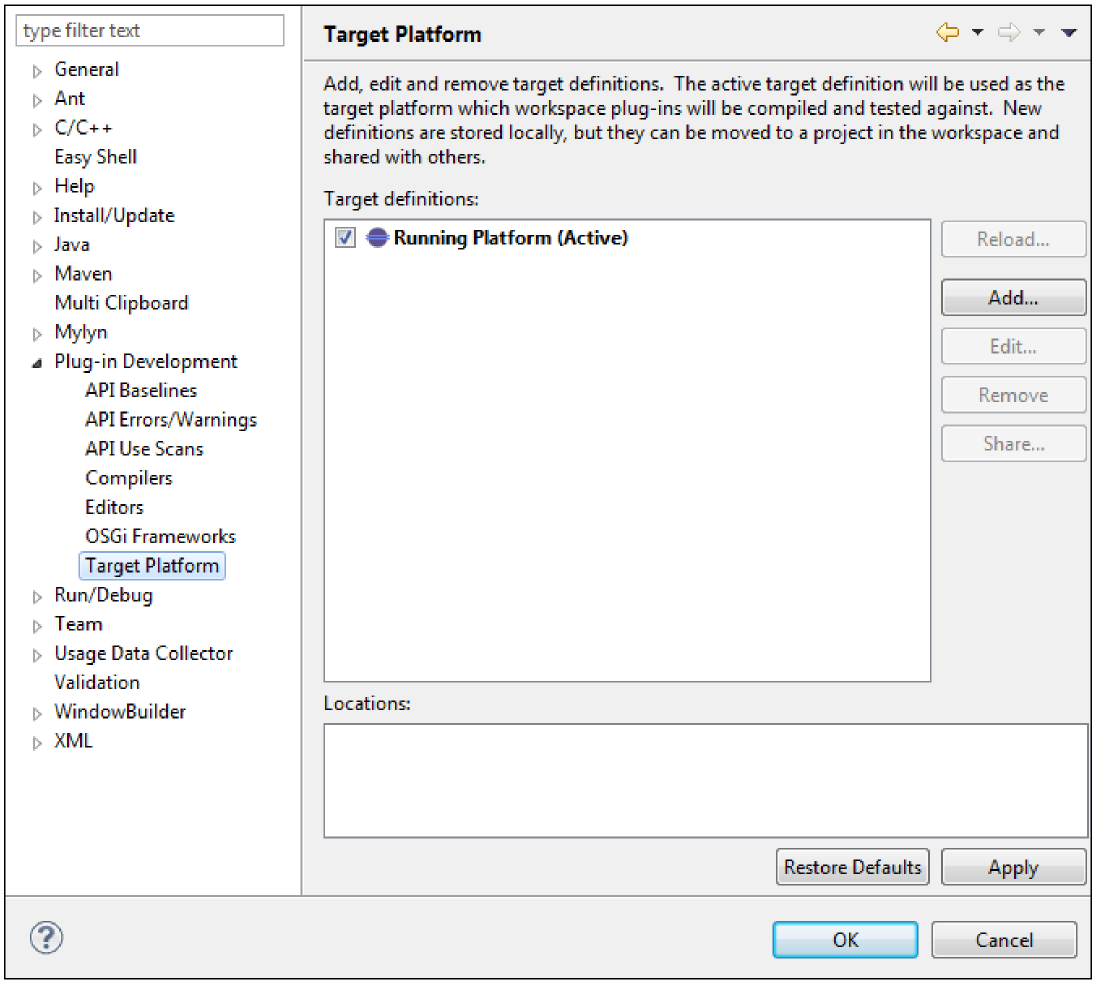
Perform the following steps to configure Eclipse and create an empty plug-in for Teamcenter.

Eclipse 3.7 SR2 was used in the examples below.

Add a new Target Platform for plug-in development, Eclipse will use this environment to check for errors/warnings in your code according to the RAC/Teamcenter Integration for CATIA V5 APIs.

1. Start Eclipse.
2. Pick **Window >> Preferences** , expand **Plug-in Development** and select **Target Platform**.

3. Pick **Add...** to create a new Target platform.



Target Platform

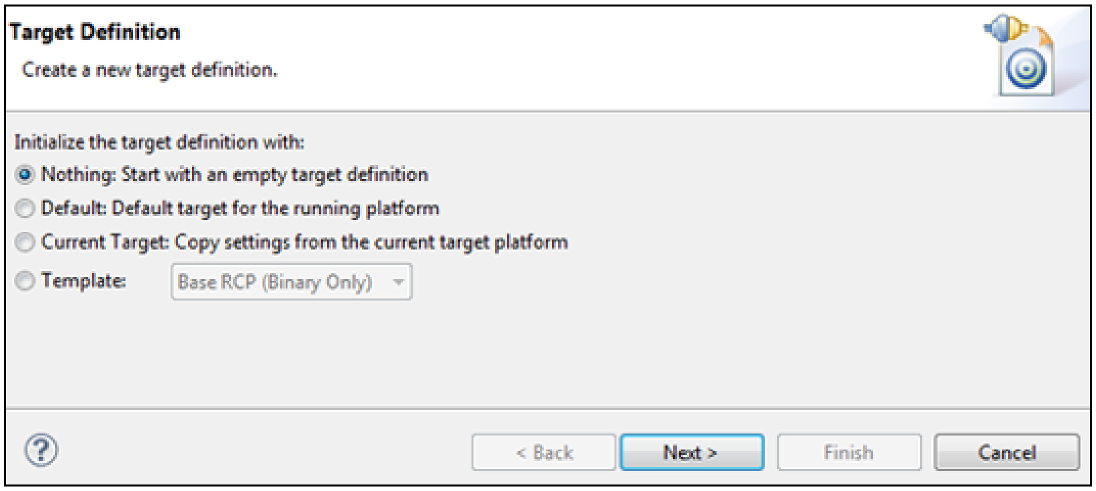
Add, edit and remove target definitions. The active target definition will be used as the target platform which workspace plug-ins will be compiled and tested against. New definitions are stored locally, but they can be moved to a project in the workspace and shared with others.

Target definitions:

- [✓] Running Platform (Active)

Locations:

4. Select **Nothing: Start with an empty target definition** then pick **Next** to continue.



Target Definition

Create a new target definition.

Initialize the target definition with:

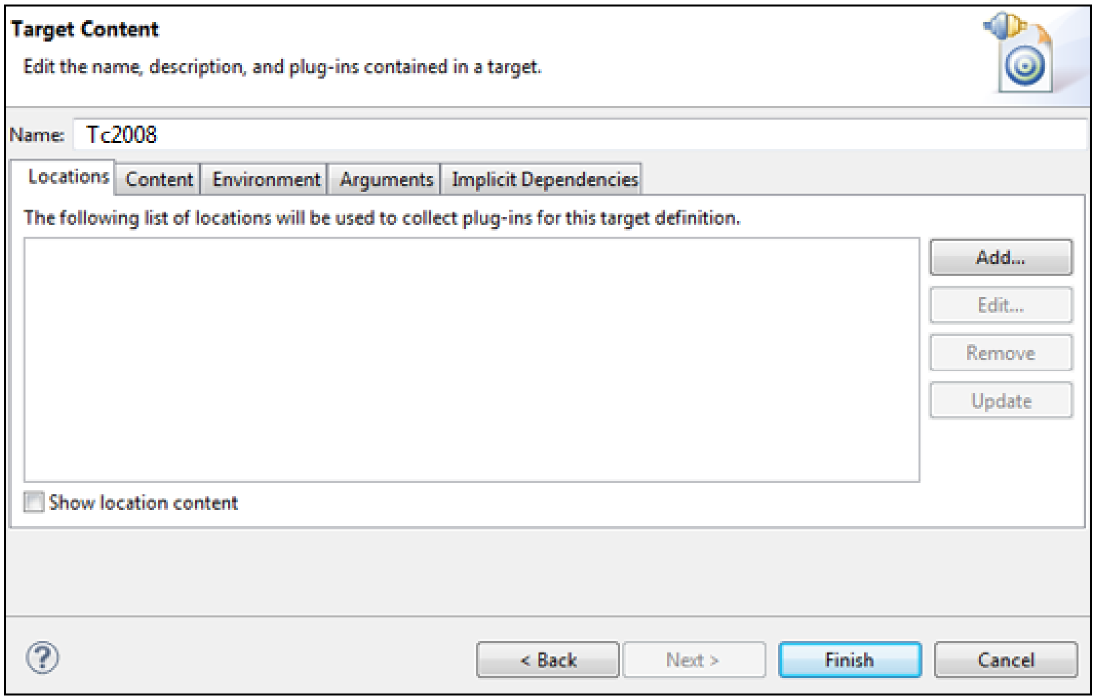
[✓ Nothing: Start with an empty target definition]

Default: Default target for the running platform

Current Target: Copy settings from the current target platform

Template: Base RCP (Binary Only)

5. Enter a Name for the environment, for example *TC2008* or *Tc9*, then pick **Add…** to localize the Teamcenter portal installation.



Target Content

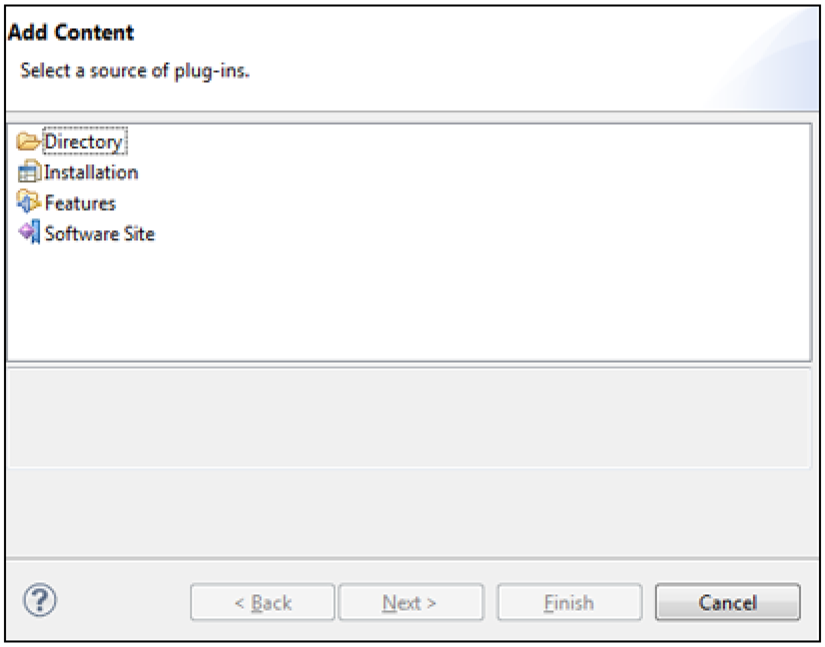
Edit the name, description, and plug-ins contained in a target.

Name: Tc2008

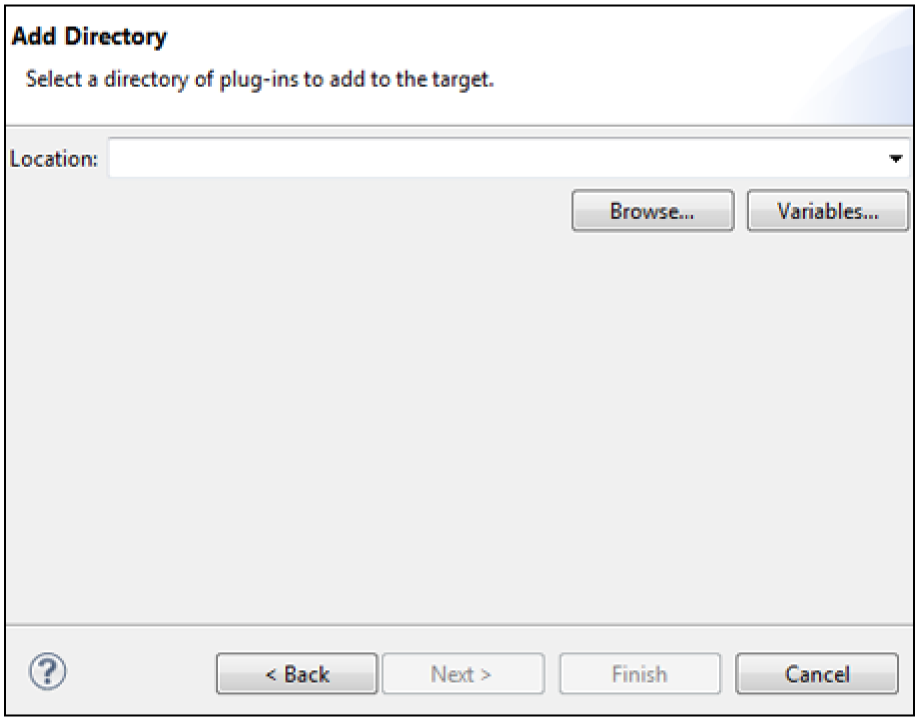
The following list of locations will be used to collect plug-ins for this target definition.

Show location content

6. Select **Directory**, then pick **Next**.

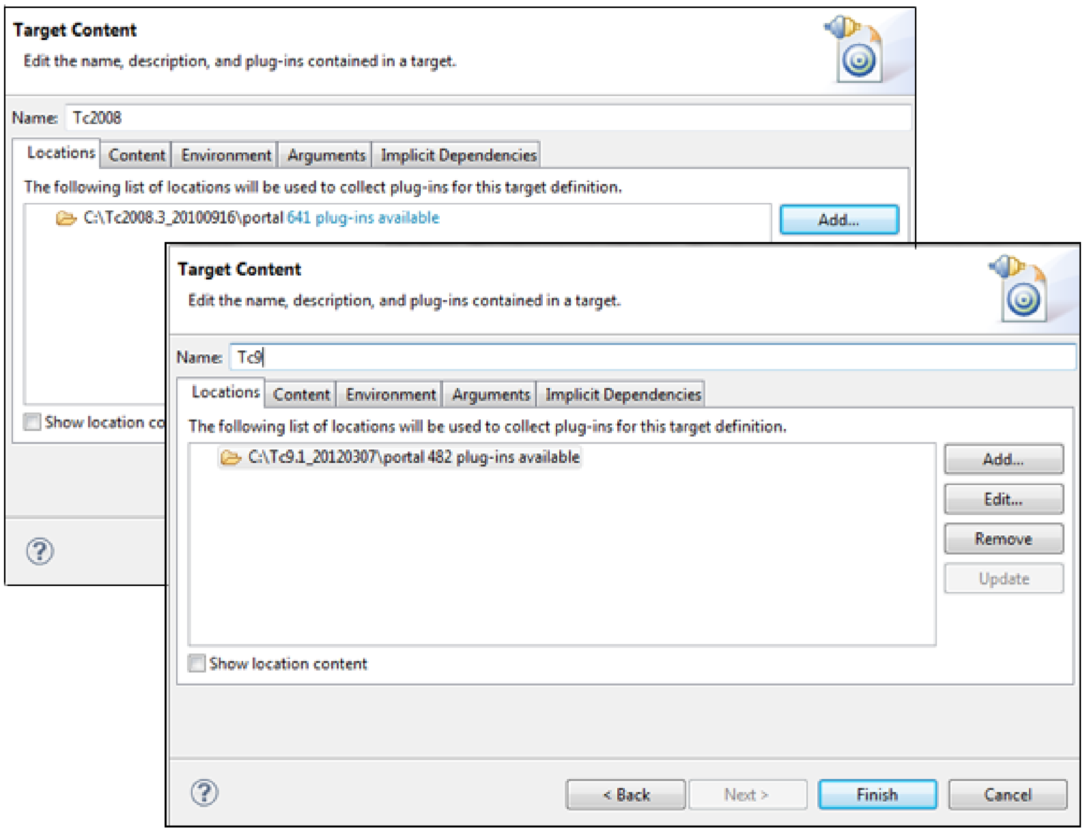


7. Pick the **Browse** button to locate the TC_PORTAL_ROOT directory, for example *C:\Tc<version>\portal.*



8. Pick **Finish** when done.

9. Please wait a moment while Eclipse searches for plug-ins. The example below shows Teamcenter 8 and 9.1 target platforms. Pick **Finish** when the process completes.



Target Content

Edit the name, description, and plug-ins contained in a target.

Name: Tc2008

The following list of locations will be used to collect plug-ins for this target definition.

C:\Tc2008.3_20100916\portal 641 plug-ins available

Target Content

Edit the name, description, and plug-ins contained in a target.

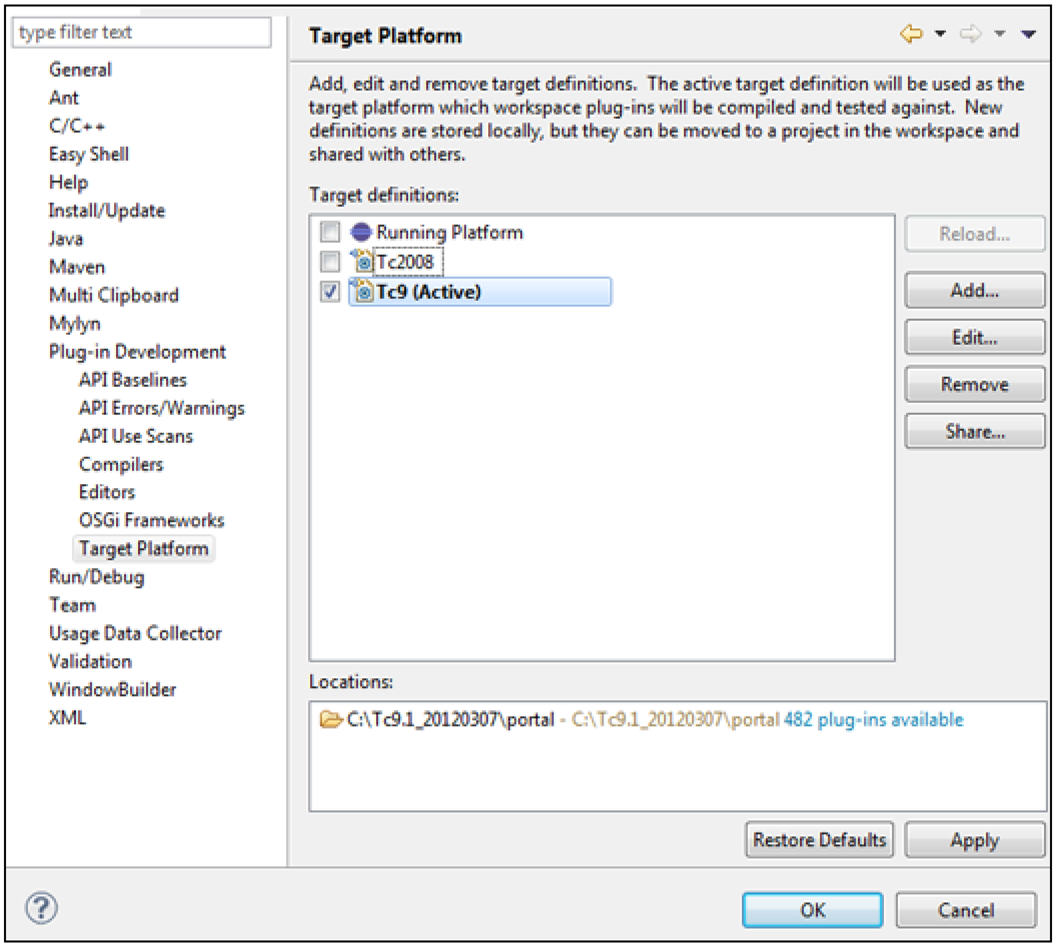
Name: Tc9

The following list of locations will be used to collect plug-ins for this target definition.

C:\Tc9.1_20120307\portal 482 plug-ins available

Show location content

10. Choose the target platform for the plug-in then pick **OK** to update the Eclipse environment. When the plug-in project is already open, Eclipse compiles it and checks it for errors.



Target Platform

Add, edit and remove target definitions. The active target definition will be used as the target platform which workspace plug-ins will be compiled and tested against. New definitions are stored locally, but they can be moved to a project in the workspace and shared with others.

Target definitions:

- [ ] Running Platform
- [ ] Tc2008
- [✓] Tc9 (Active)

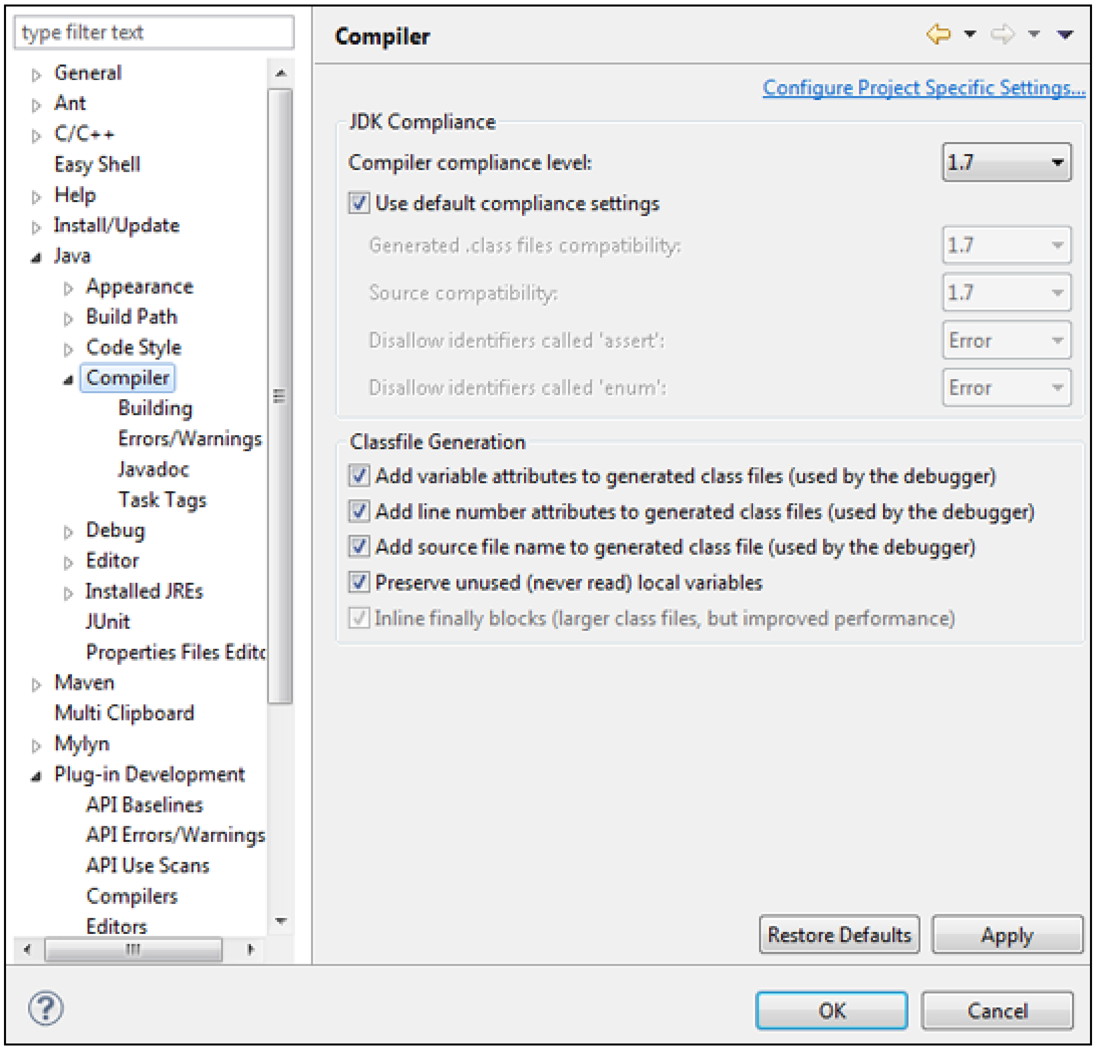
Locations:

C:\Tc9.1_20120307\portal - C:\Tc9.1_20120307\portal 482 plug-ins available

Configure the JAVA Compiler

The steps below use the Teamcenter 9.1 platform.

11. Pick **Window >> Preferences** then expand **Java** and select **Compiler**.



JDK Compliance

Compiler compliance level: 1.7

[✓] Use default compliance settings

Generated .class files compatibility: 1.7

Source compatibility: 1.7

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

[✓] Add variable attributes to generated class files (used by the debugger)

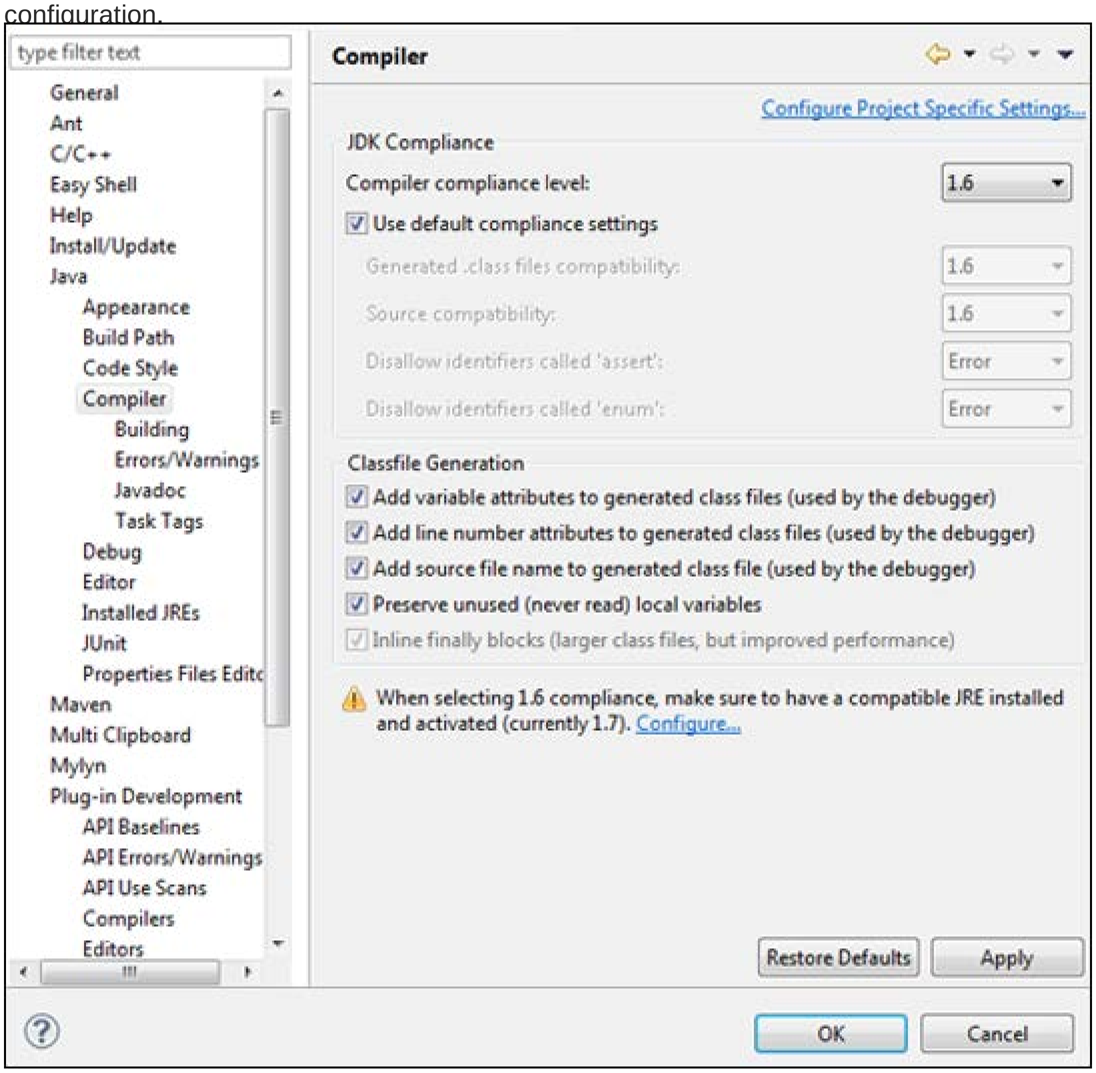
[✓] Add line number attributes to generated class files (used by the debugger)

[✓] Add source file name to generated class file (used by the debugger)

[✓] Preserve unused (never read) local variables

[✓] Inline finally blocks (larger class files, but improved performance)

12. Select a **Compiler compliance level**. In this example, the Integration plug-ins were compiled with Java JDK 6 so the compliance level chosen is **1.6**. Pick **OK** to complete the configuration.



JDK Compliance

Compiler compliance level: 1.6

☑ Use default compliance settings

Generated .class files compatibility: 1.6

Source compatibility: 1.6

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

☑ Add variable attributes to generated class files (used by the debugger)

☑ Add line number attributes to generated class files (used by the debugger)

☑ Add source file name to generated class file (used by the debugger)

☑ Preserve unused (never read) local variables

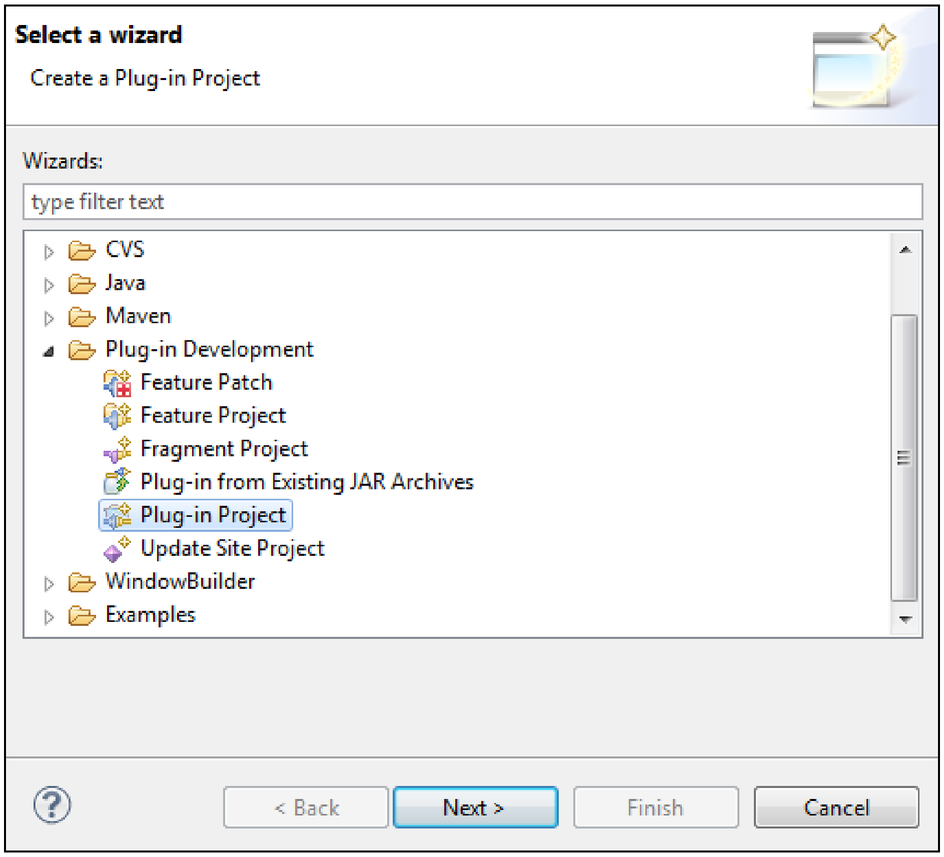
☑ Inline finally blocks (larger class files, but improved performance)

When selecting 1.6 compliance, make sure to have a compatible JRE installed and activated (currently 1.7). Configure...

7.3.5.2.2 *Create the Plug-in Project*

1. Pick **File >> New Project**.

2. Open **Plug-in Development** then select **Plug-in Project** from the New Project window, then pick **Next** to continue.


Select a wizard

Create a Plug-in Project

Wizards:

type filter text

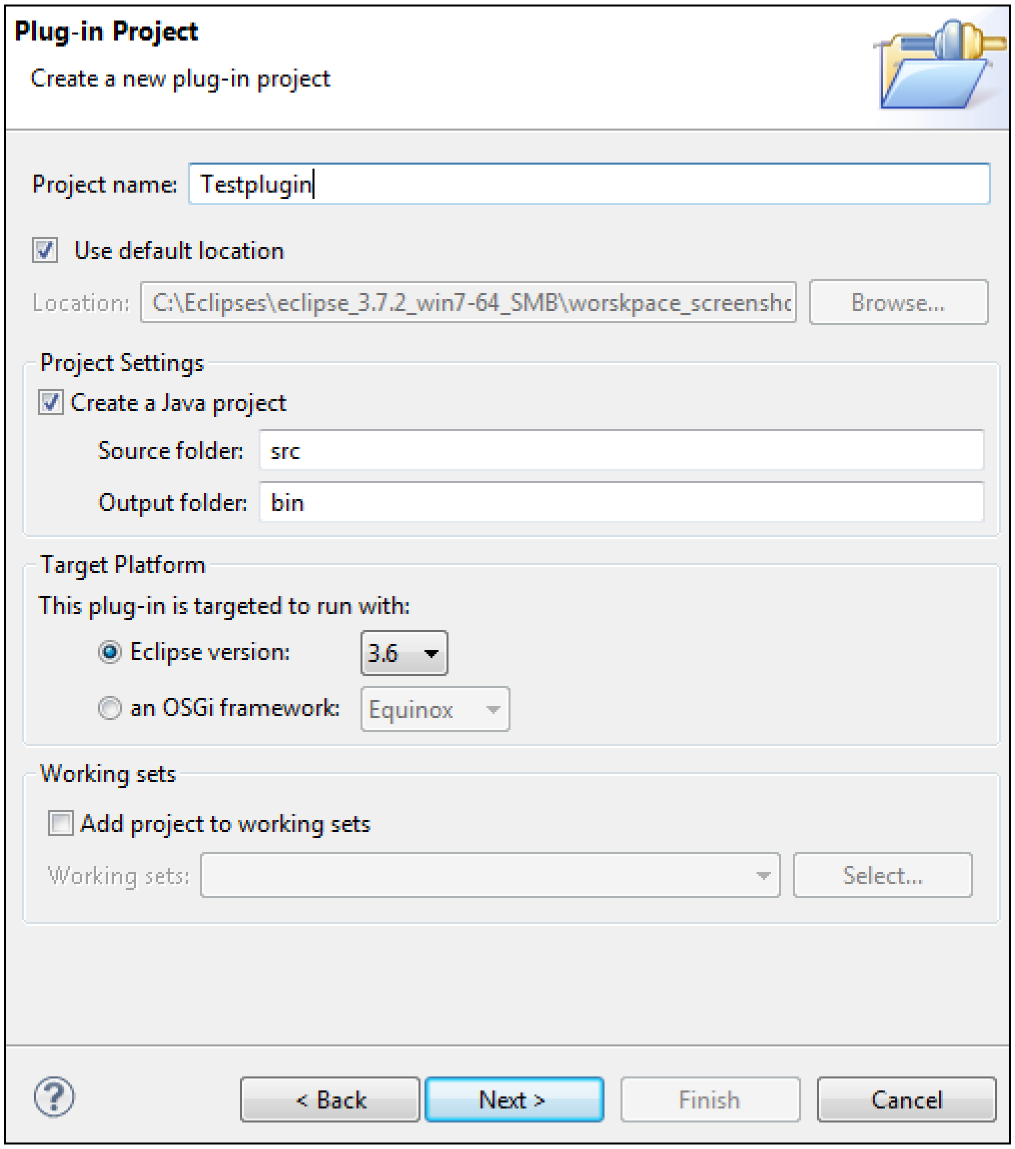
- CVS
- Java
- Maven
- Plug-in Development
  - Feature Patch
  - Feature Project
  - Fragment Project
  - Plug-in from Existing JAR Archives
  - Plug-in Project
  - Update Site Project
- WindowBuilder
- Examples

3. Complete the Plug-in Project screen, then pick **Next** to continue.

   a. Enter a project name in the Project Name: field.

   b. Toggle **Create a Java project** on, then complete the Source folder: and Output folder: fields.

   c. Toggle **Eclipse version** on, then select the appropriate Eclipse version for example, select **3.5** for Teamcenter 2008.3: and **3.6** for Teamcenter 9.


Plug-in Project

Create a new plug-in project

Project name: Testplugin

[✓] Use default location

Location: C:\Eclipses\eclipse_3.7.2_win7-64_SMB\worskpace_screenshc

Project Settings

[✓] Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

(●) Eclipse version: 3.6

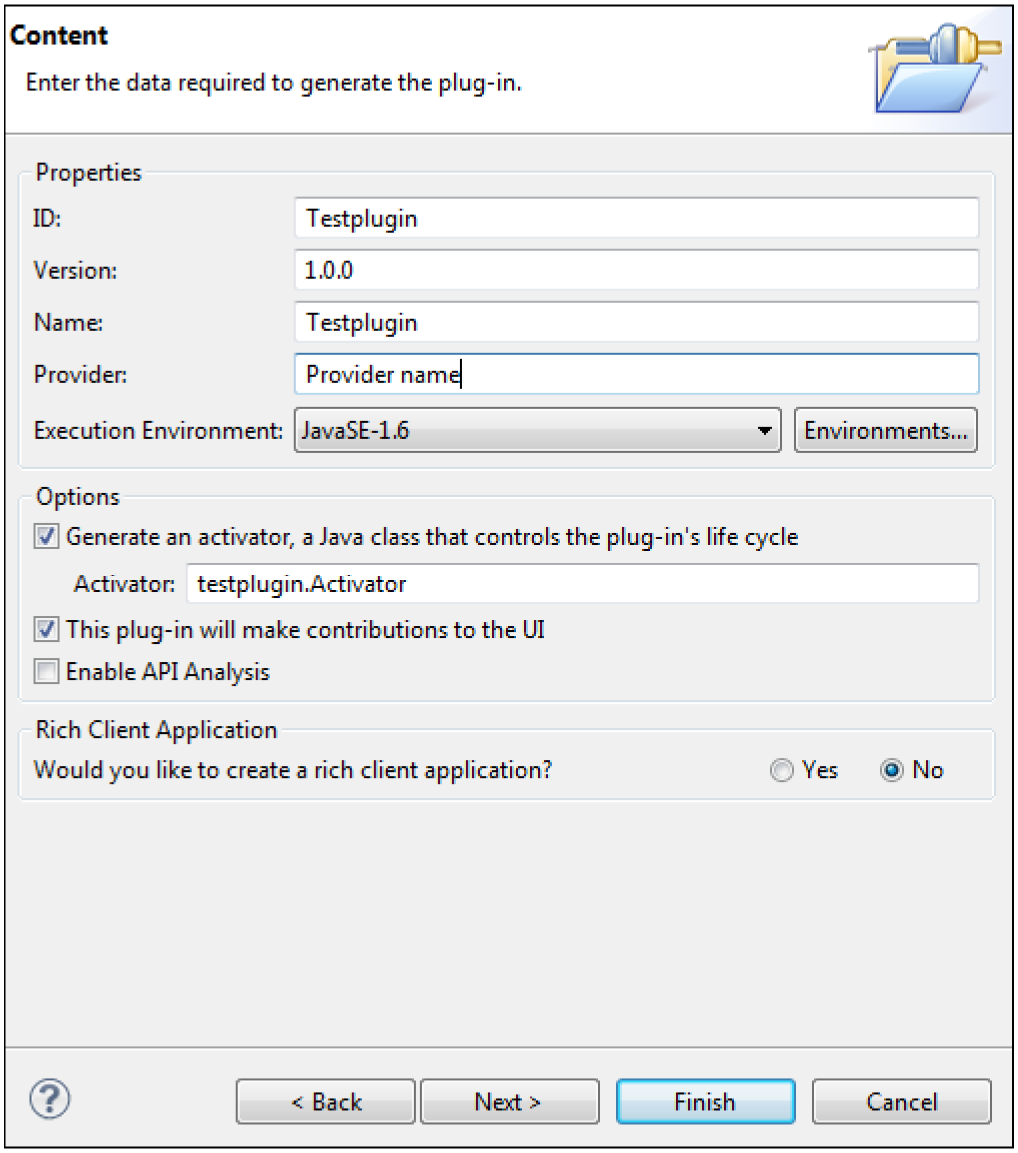
( ) an OSGi framework: Equinox

Working sets

[ ] Add project to working sets

Working sets:

4. Complete the plug-in ID,Version and Name fields on the Plug-in Content screen, then pick **Next** to continue. Teamcenter references the plug-in defined in the Name field.


Content

Enter the data required to generate the plug-in.

Properties

ID: Testplugin

Version: 1.0.0

Name: Testplugin

Provider: Provider name

Execution Environment: JavaSE-1.6

Options

☑ Generate an activator, a Java class that controls the plug-in's life cycle

Activator: testplugin.Activator

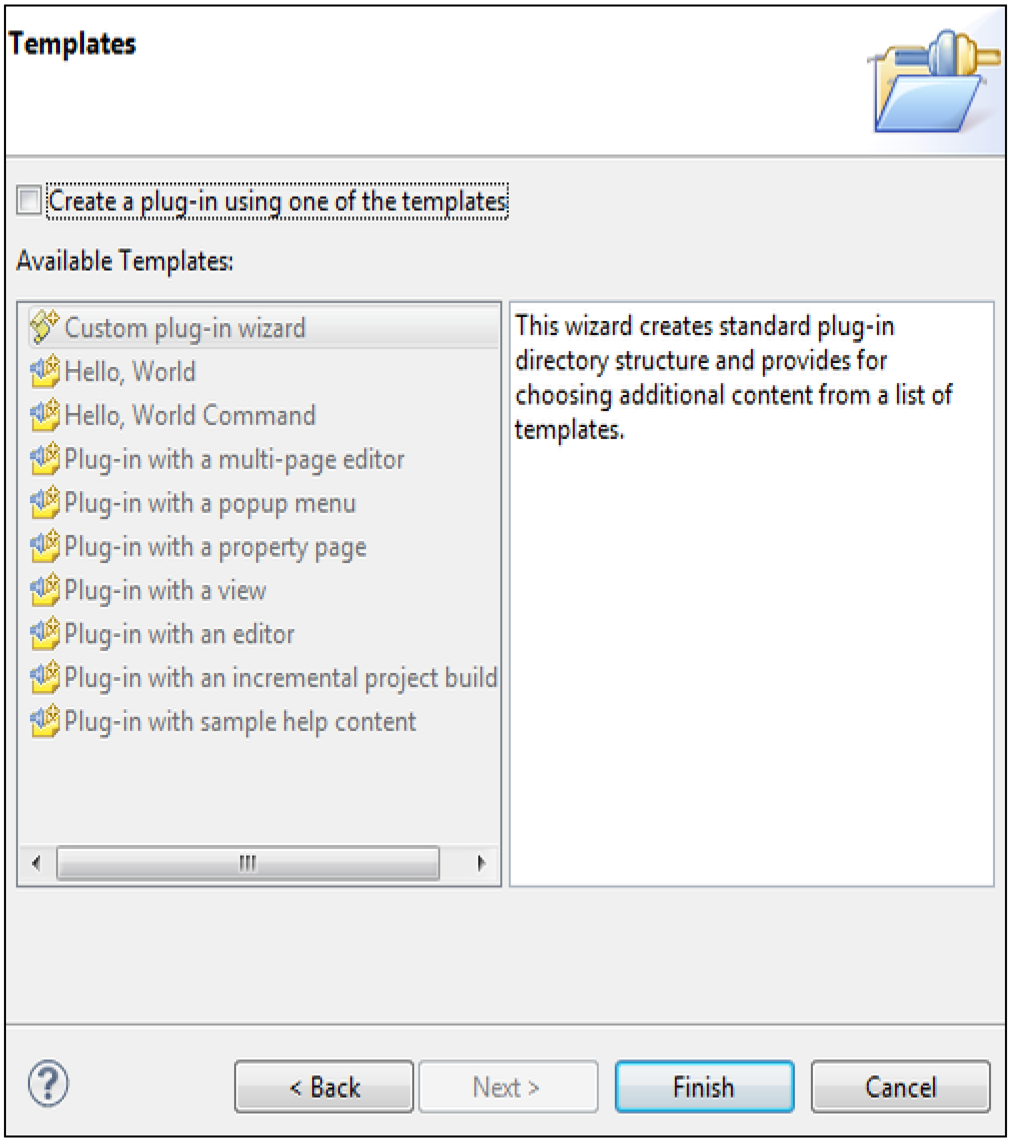
☑ This plug-in will make contributions to the UI

☐ Enable API Analysis

Rich Client Application

Would you like to create a rich client application? Yes [✓ No]

5. Toggle **Create a plug-in using one of the templates** off to create an empty plug-in. Select a Teamcenter plug-in template from the Templates screen. Pick **Finish** when done.



Templates

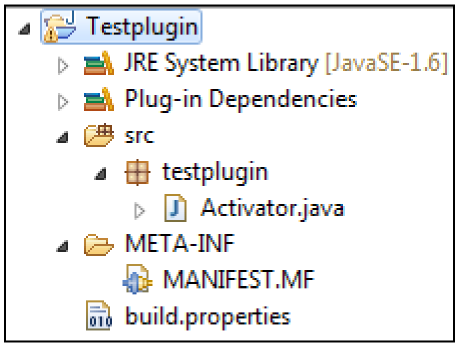
[ ] Create a plug-in using one of the templates

Available Templates:

- Custom plug-in wizard
- Hello, World
- Hello, World Command
- Plug-in with a multi-page editor
- Plug-in with a popup menu
- Plug-in with a property page
- Plug-in with a view
- Plug-in with an editor
- Plug-in with an incremental project build
- Plug-in with sample help content

This wizard creates standard plug-in directory structure and provides for choosing additional content from a list of templates.

6. In Eclipse, verify the plug-in was created.



```
Testplugin
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src
    testplugin
      Activator.java
  META-INF
    MANIFEST.MF
  build.properties
```

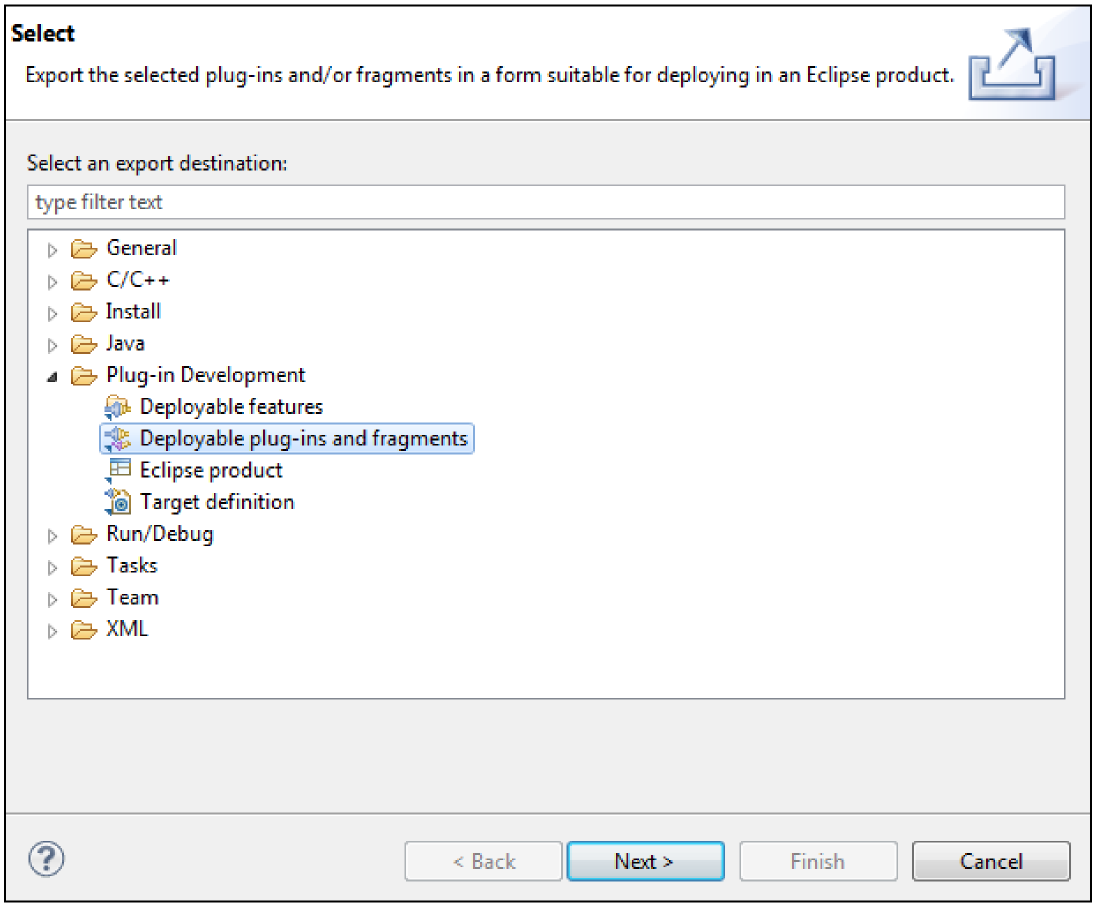
7.3.5.2.3 *Export a Plug-in into Teamcenter*

Perform the following steps to export a plug-in into Teamcenter.

1. In Eclipse, select the plug-in project.

2. Pick **File >> Export...**.

3. Expand **Plug-in Development** then select **Deployable plug-ins and fragments**. Pick **Next** to continue.



Select

Export the selected plug-ins and/or fragments in a form suitable for deploying in an Eclipse product.

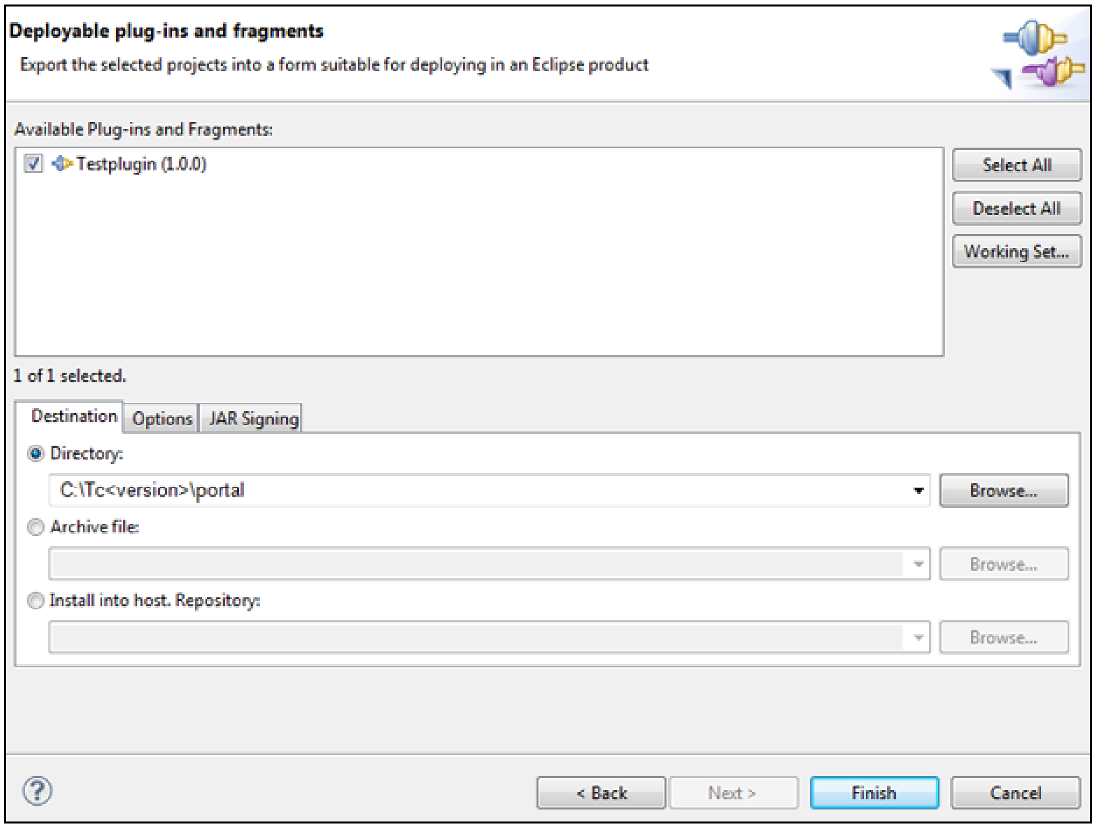
Select an export destination:

type filter text

4. Select **TestPlugin (1.0.0)** from the Deployable plug-ins and fragments screen.

5. Complete the Directory field within the Destination tab. For example, enter *C:\Tc<version>\portal.*

6. Pick **Finish** to export the plug-in



Deployable plug-ins and fragments

Export the selected projects into a form suitable for deploying in an Eclipse product

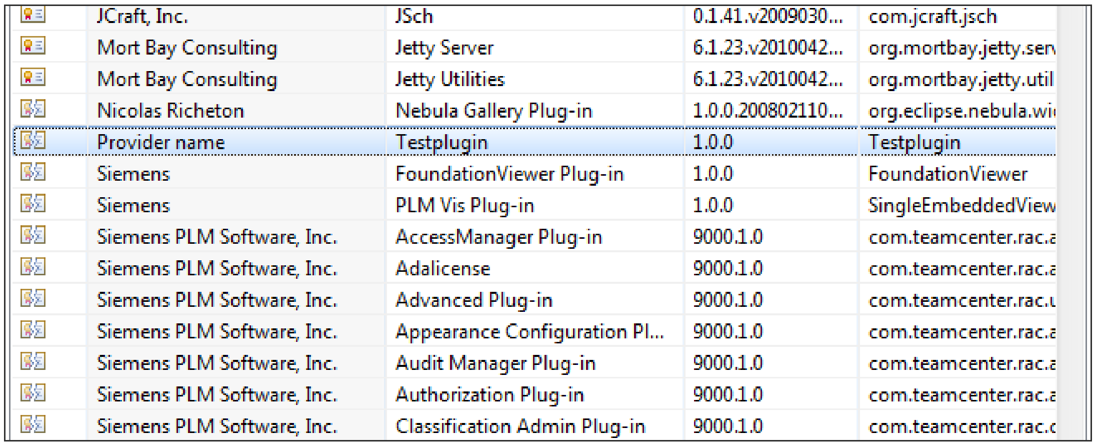
Available Plug-ins and Fragments:

[✓] Testplugin (1.0.0)

1 of 1 selected.

Directory: C:\Tc<version>\portal

7. Pick **Help >> About** then review the Installation Details screen to ensure the plug-in has been accepted.

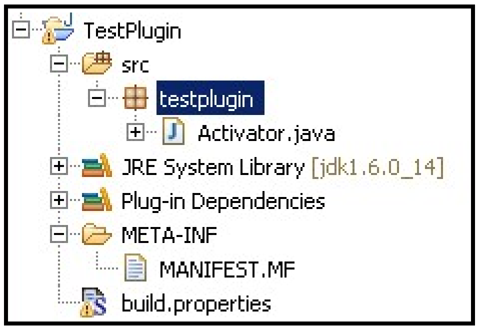


JCraft, Inc.	JSch	0.1.41.v2009030...	com.jcraft.jsch
Mort Bay Consulting	Jetty Server	6.1.23.v2010042...	org.mortbay.jetty.serv
Mort Bay Consulting	Jetty Utilities	6.1.23.v2010042...	org.mortbay.jetty.util
Nicolas Richeton	Nebula Gallery Plug-in	1.0.0.200802110...	org.eclipse.nebula.wi
Provider name	Testplugin	1.0.0	Testplugin
Siemens	FoundationViewer Plug-in	1.0.0	FoundationViewer
Siemens	PLM Vis Plug-in	1.0.0	SingleEmbeddedView
Siemens PLM Software, Inc.	AccessManager Plug-in	9000.1.0	com.teamcenter.rac.a
Siemens PLM Software, Inc.	Adalicense	9000.1.0	com.teamcenter.rac.a
Siemens PLM Software, Inc.	Advanced Plug-in	9000.1.0	com.teamcenter.rac.u
Siemens PLM Software, Inc.	Appearance Configuration Pl...	9000.1.0	com.teamcenter.rac.a
Siemens PLM Software, Inc.	Audit Manager Plug-in	9000.1.0	com.teamcenter.rac.a
Siemens PLM Software, Inc.	Authorization Plug-in	9000.1.0	com.teamcenter.rac.a
Siemens PLM Software, Inc.	Classification Admin Plug-in	9000.1.0	com.teamcenter.rac.c

7.3.5.2.4 Example: Customize a Save Manager Button

Use the following example to modify the behavior of the Save button in the Save Manager so that it displays a message when selected.

1. Create an empty plug-in. For example,



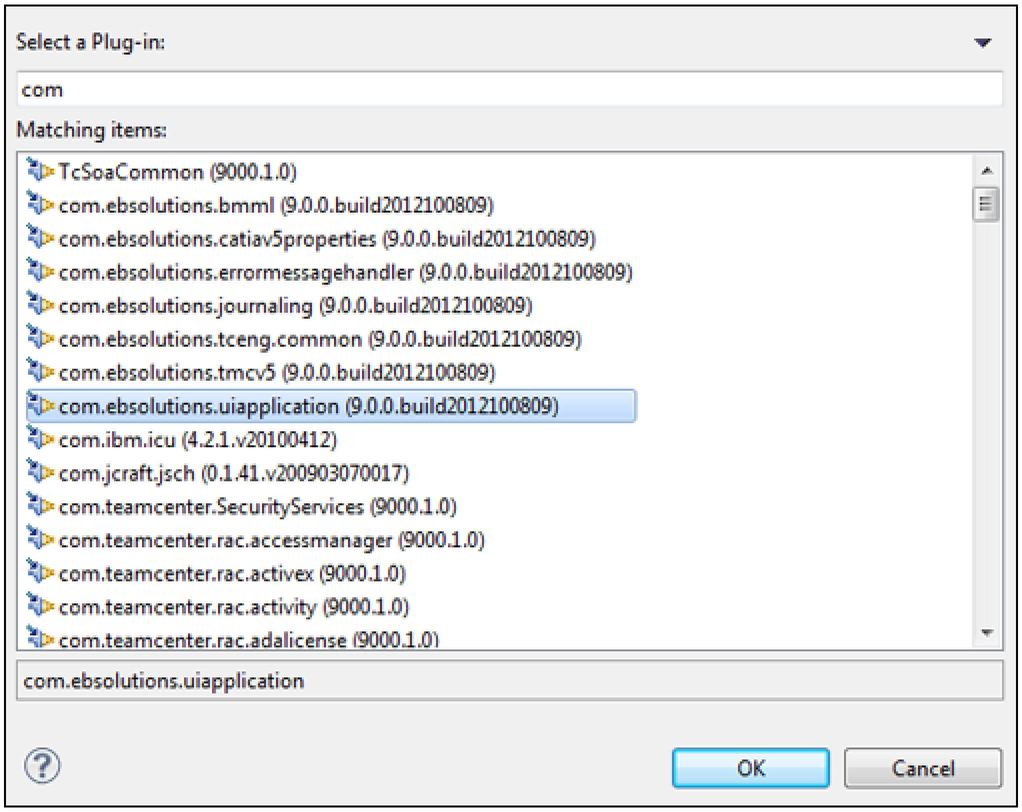
```
TestPlugin
  src
    testplugin
      Activator.java
  JRE System Library [jdk1.6.0_14]
  Plug-in Dependencies
  META-INF
    MANIFEST.MF
  build.properties
```

2. Edit the *MANIFEST.MF* file.

   a. Select the **Dependencies** tab in the Eclipse PDE.

   b. Pick **Add...** and select the uiapplication plug-in indicating that *cat2uimanager.jar* file must be run.

c. Pick **OK** when finished.



Select a Plug-in:

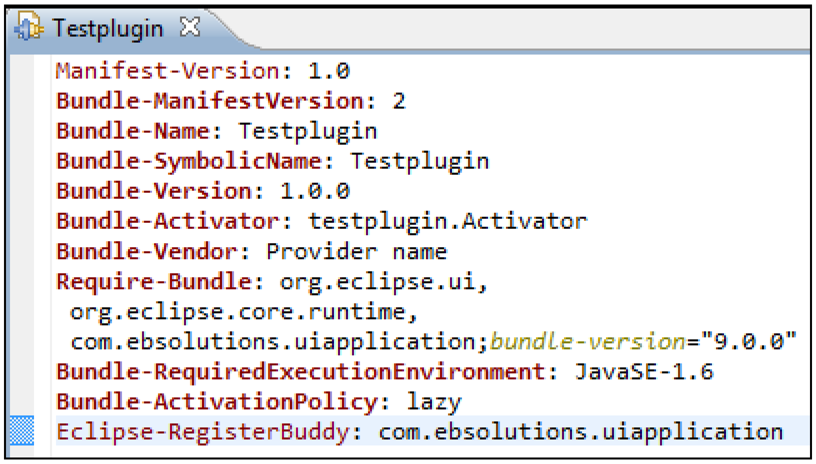
com

Matching items:

```
TcSoaCommon (9000.1.0)
com.ebsolutions.bmml (9.0.0.build2012100809)
com.ebsolutions.catiav5properties (9.0.0.build2012100809)
com.ebsolutions.errormessagehandler (9.0.0.build2012100809)
com.ebsolutions.journaling (9.0.0.build2012100809)
com.ebsolutions.tceng.common (9.0.0.build2012100809)
com.ebsolutions.tmcv5 (9.0.0.build2012100809)
com.ebsolutions.uiapplication (9.0.0.build2012100809)
com.ibm.icu (4.2.1.v20100412)
com.jcraft.jsch (0.1.41.v200903070017)
com.teamcenter.SecurityServices (9000.1.0)
com.teamcenter.rac.accessmanager (9000.1.0)
com.teamcenter.rac.activex (9000.1.0)
com.teamcenter.rac.activity (9000.1.0)
com.teamcenter.rac.adalicense (9000.1.0)
```

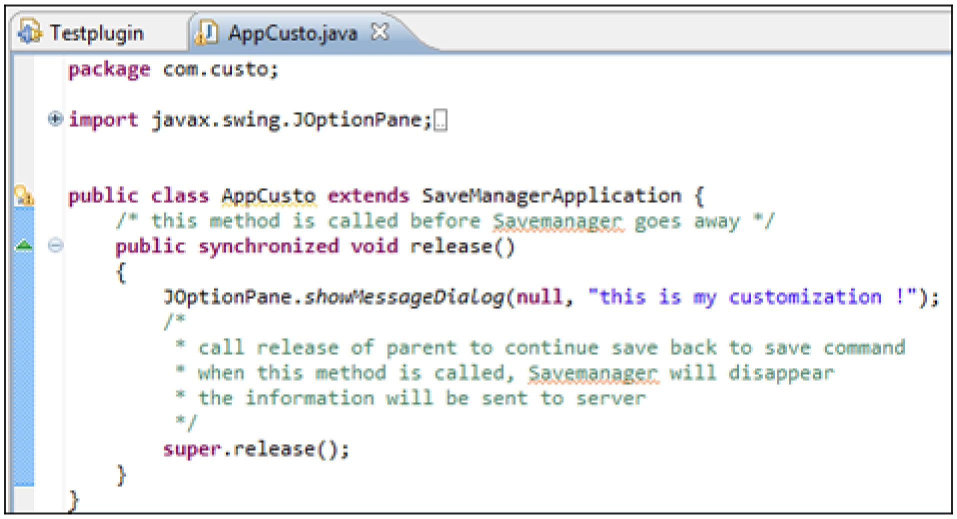
com.ebsolutions.uiapplication

3. Select the **MANIFEST.MF** tab then pick **OK** to add the SymblocName of the plug-ins under Require-Bundle.



```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Testplugin
Bundle-SymbolicName: Testplugin
Bundle-Version: 1.0.0
Bundle-Activator: testplugin.Activator
Bundle-Vendor: Provider name
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime,
 com.ebsolutions.uiapplication;bundle-version="9.0.0"
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Bundle-ActivationPolicy: lazy
Eclipse-RegisterBuddy: com.ebsolutions.uiapplication
```

4. Create a package named *com.custo*, then add a class called *AppCusto.java*.



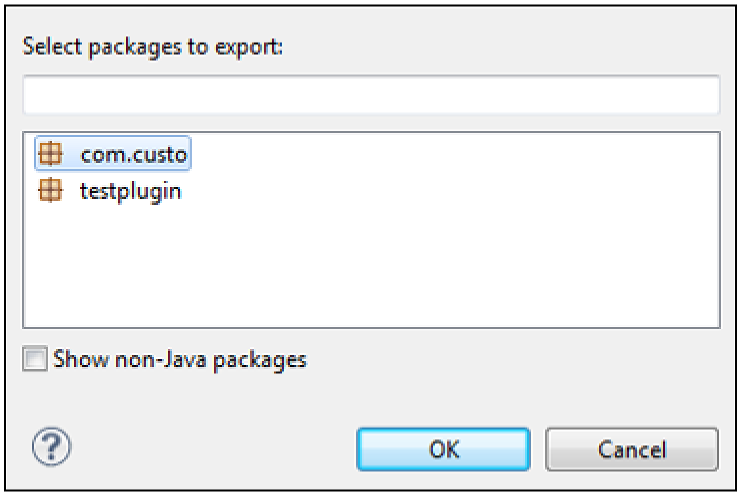
```
package com.custo;

import javax.swing.JOptionPane;

public class AppCusto extends SaveManagerApplication {
    /* this method is called before Savemanager goes away */
    public synchronized void release()
    {
        JOptionPane.showMessageDialog(null, "this is my customization !");
        /*
         * call release of parent to continue save back to save command
         * when this method is called, Savemanager will disappear
         * the information will be sent to server
         */
        super.release();
    }
}
```

5. Make the new package accessible from the Integration's Cat2tciccommon.jar plug-in.

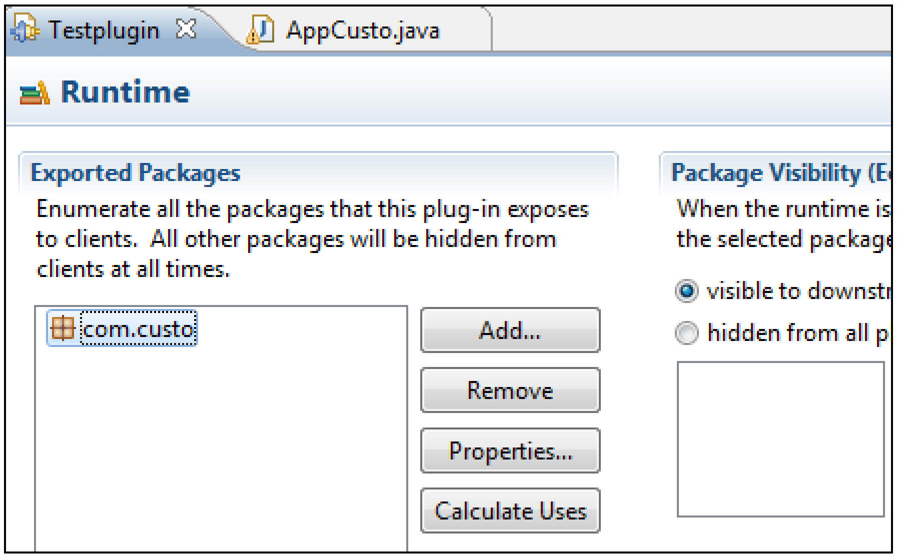
   a. Select the **Runtime** tab in the Eclipse PDE.

   b. Pick **Add...**

   c. Select the **com.custo** package, then pick **OK**.



6. Export the package.



7. Delete the RAC directory and launch the Teamcenter *genregxml.bat*, located in the *<Portal directory>\registry*.

8. Open the *TcIC_DIR\env\Savemanager.xml* file and change the Save Manager application as shown below.

```
replace:
<UIApplication class="com.ebsolutions.savemanager.SaveManagerApplication"
             nameRef="Teamcenter Save Manager"
             image="/com/ebsolutions/uimanager/modules/images/savemanager.png"
             sizex="1000" sizey="600" lang="">

with:
<UIApplication class="com.custo.AppCusto"
             nameRef="My Customization"
             image="/com/ebsolutions/uimanager/modules/images/savemanager.png"
             sizex="1000" sizey="600" lang="">
```

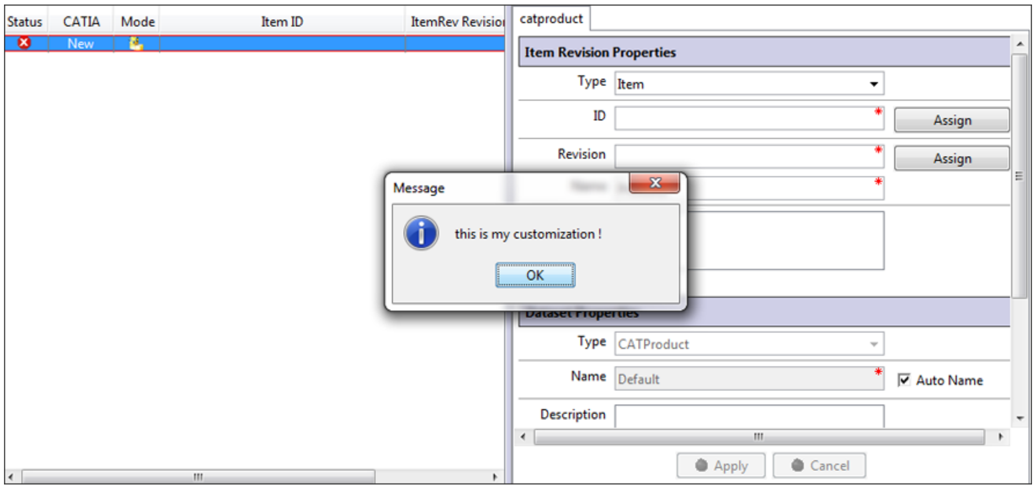
9. Test the customization.

   a. Start CATIA and Teamcenter.

   b. Verify the plug-in has been accepted by Teamcenter.

   c. In CATIA, create A1(P1), then **Save**.

   d. In the Save Manager, assign IDs, then pick the **Save** button. The new message displays.

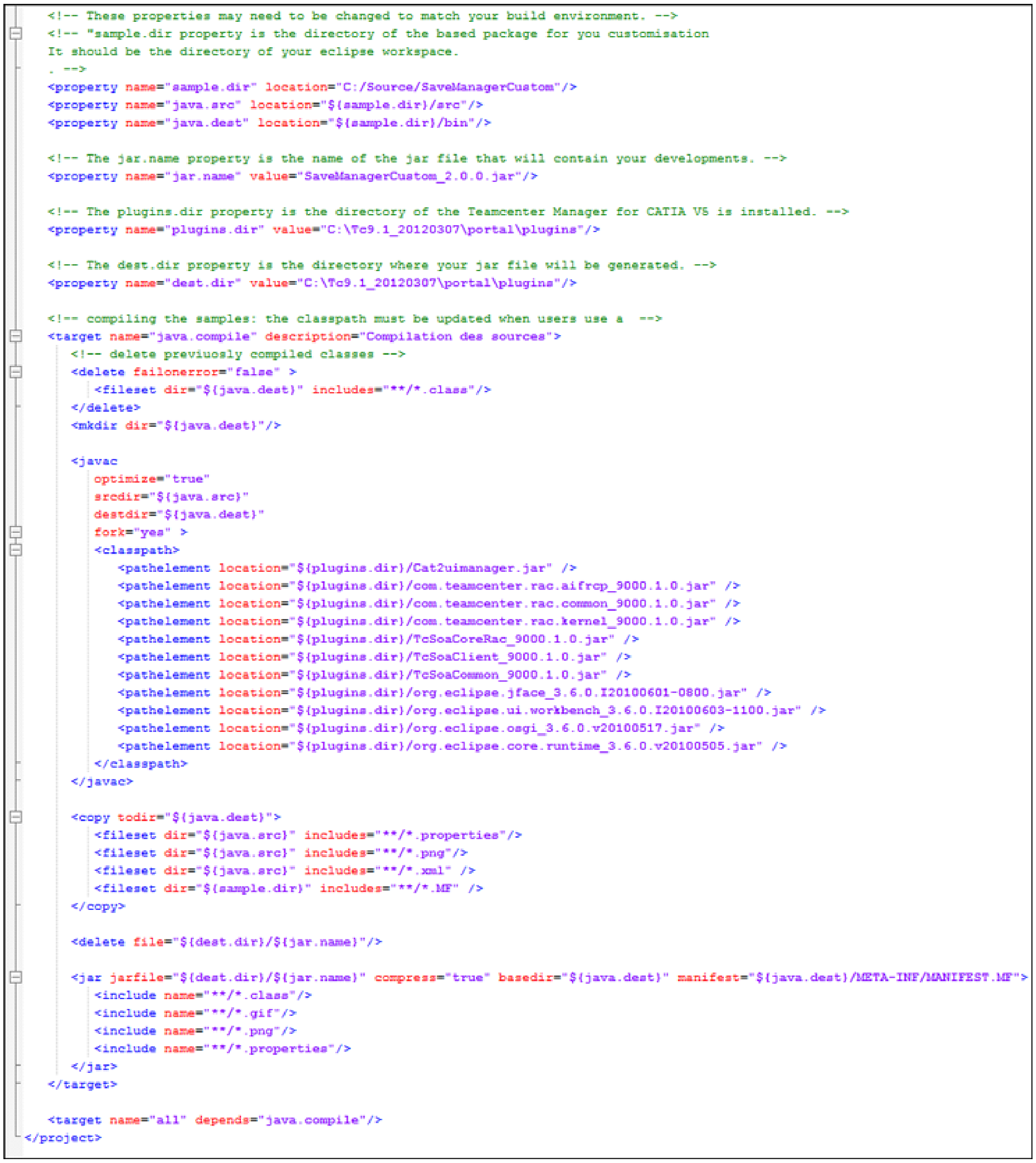


this is my customization !

OK

7.3.5.2.5 *Creating an Ant Build Script*

Use an *Ant script* within Eclipse to compile and create a .jar file containing your customization. This Ant script can also be used for testing. Refer to the following build example. Adapt these instructions to include paths specific to your needs.



```
    <!-- These properties may need to be changed to match your build environment. -->
    <!-- "sample.dir property is the directory of the based package for you customisation
    It should be the directory of your eclipse workspace.
    . -->
    <property name="sample.dir" location="C:/Source/SaveManagerCustom"/>
    <property name="java.src" location="${sample.dir}/src"/>
    <property name="java.dest" location="${sample.dir}/bin"/>

    <!-- The jar.name property is the name of the jar file that will contain your developments. -->
    <property name="jar.name" value="SaveManagerCustom_2.0.0.jar"/>

    <!-- The plugins.dir property is the directory of the Teamcenter Manager for CATIA V5 is installed. -->
    <property name="plugins.dir" value="C:\Tc9.1_20120307\portal\plugins"/>

    <!-- The dest.dir property is the directory where your jar file will be generated. -->
    <property name="dest.dir" value="C:\Tc9.1_20120307\portal\plugins"/>

    <!-- compiling the samples: the classpath must be updated when users use a  -->
    <target name="java.compile" description="Compilation des sources">
       <!-- delete previuosly compiled classes -->
       <delete failonerror="false" >
          <fileset dir="${java.dest}" includes="**/*.class"/>
       </delete>
       <mkdir dir="${java.dest}"/>

       <javac
          optimize="true"
          srcdir="${java.src}"
          destdir="${java.dest}"
          fork="yes" >
          <classpath>
             <pathelement location="${plugins.dir}/Cat2uimanager.jar" />
             <pathelement location="${plugins.dir}/com.teamcenter.rac.aifrcp_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/com.teamcenter.rac.common_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/com.teamcenter.rac.kernel_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/TcSoaCoreRac_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/TcSoaClient_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/TcSoaCommon_9000.1.0.jar" />
             <pathelement location="${plugins.dir}/org.eclipse.jface_3.6.0.I20100601-0800.jar" />
             <pathelement location="${plugins.dir}/org.eclipse.ui.workbench_3.6.0.I20100603-1100.jar" />
             <pathelement location="${plugins.dir}/org.eclipse.osgi_3.6.0.v20100517.jar" />
             <pathelement location="${plugins.dir}/org.eclipse.core.runtime_3.6.0.v20100505.jar" />
          </classpath>
       </javac>

       <copy todir="${java.dest}">
          <fileset dir="${java.src}" includes="**/*.properties"/>
          <fileset dir="${java.src}" includes="**/*.png"/>
          <fileset dir="${java.src}" includes="**/*.xml" />
          <fileset dir="${sample.dir}" includes="**/*.MF" />
       </copy>

       <delete file="${dest.dir}/${jar.name}"/>

       <jar jarfile="${dest.dir}/${jar.name}" compress="true" basedir="${java.dest}" manifest="${java.dest}/META-INF/MANIFEST.MF">
          <include name="**/*.class"/>
          <include name="**/*.gif"/>
          <include name="**/*.png"/>
          <include name="**/*.properties"/>
       </jar>
    </target>

    <target name="all" depends="java.compile"/>
</project>
```

1. Right-click on the Java project, then pick **New >> Folder** to create a build directory within your Java project. Name the folder as required (example: *build*).

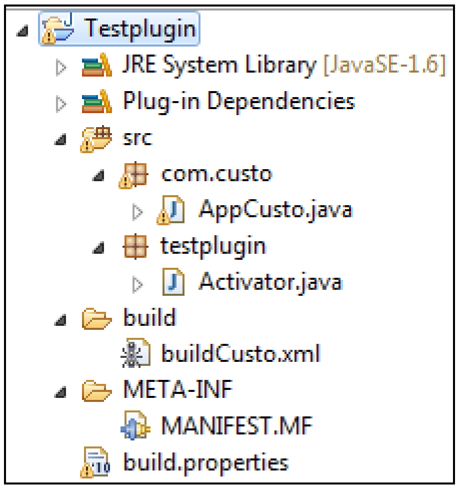
2. Right-click on the newly created directory (*build*), then pick **New >> File** to create a new file inside the directory. Name the file as required (example: *buildCusto.xml*).

3. Right-click on the file (*buildCusto.xml*) then pick **Open With >> Ant Editor** to use the Ant Editor to open the new file.

4. Edit the file as necessary. The build example shown is contained in the *buildCusto.xml* file located in the *TcIC_DIR\custom_client\SaveManager* folder.

Modify the properties at the beginning of the buildCusto.xml document specific to your required paths.

5. Ensure the project structure resembles the illustration below. NOTE: The packaged and classes, as displayed below, are for illustrative purposes and may not be found within your project.



```
Testplugin
  JRE System Library [JavaSE-1.6]
  Plug-in Dependencies
  src
    com.custo
      AppCusto.java
    testplugin
      Activator.java
  build
    buildCusto.xml
  META-INF
    MANIFEST.MF
  build.properties
```

6. Right-click on the .xml file (*buildCusto.xml*) file, then pick **Run As >> Ant Build** to compile your source and build the .jar file corresponding to the developments in the des.dir property (containing the *buildCusto.xml* file).

7.3.6 Save Manager Tutorial

Import a Project into Eclipse

Create:

Display Element
New Action
New Application
New Button Style
New Module
New Toolbar Style

Manage:

Manage CATIAdata Before/After the Save Manager Displays

Change:

Change the Table Display

APIs:

Save Manager API Java Documentation

Sample projects are available in the *projectSamples.zip* file within the *TcIC_DIR\custom_client\SaveManager* folder. Use these samples as a starting point for your customizations.

Please ensure the following when using the *projectSamples.zip* file.

- The provided SaveManagerCustom_3.0.0.jar is compatible with the current Teamcenter version (TC_VERSION) installed.
- Use the SaveManagerCustom_3.0.0.jar available under a different Tcxxx directory to be compatible with a different TC_VERSION supported by the Integration.
- The *projectSamples.zip file* has been unzipped into a directory and has been imported into Eclipse.

If necessary, refer to the **compiled** SaveManagerCustom_3.0.0.jar file, in the *TcIC_DIR\custom_client\SaveManager* folder, corresponding to the project samples, if difficulties are experienced when generating the SaveManagerCustom_3.0.0.jar file or to view the customizations.

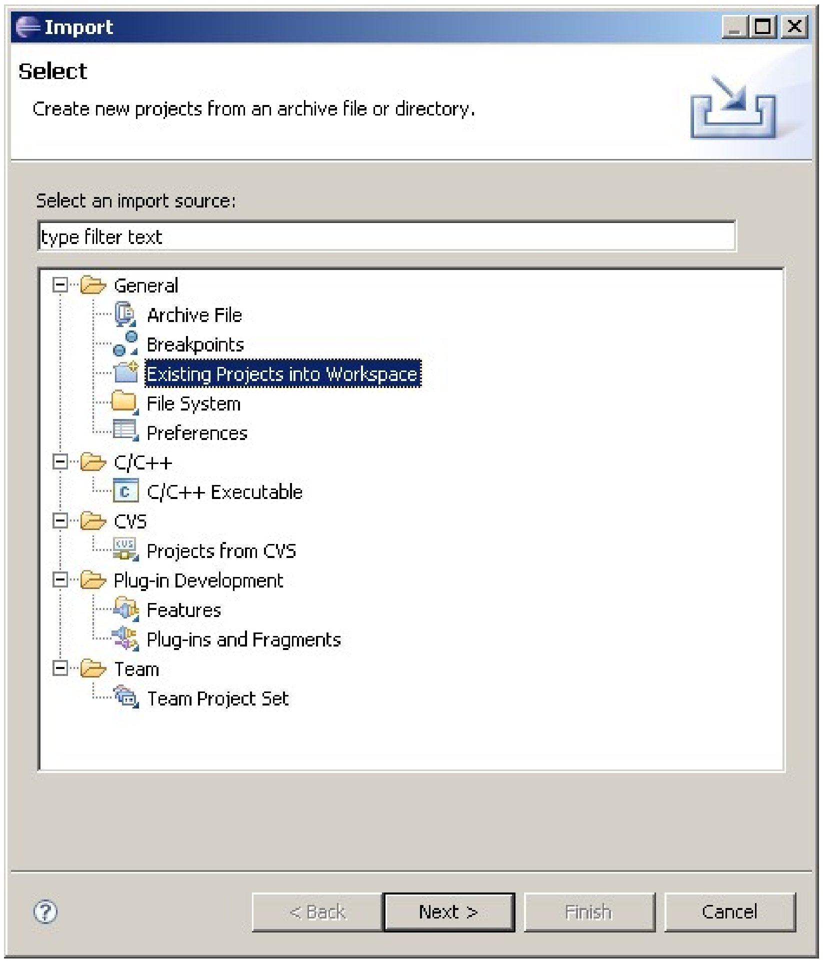
7.3.6.1 Import a Project into Eclipse

Perform the following steps to import a project into Eclipse. This example refers to the *SaveManagerCustom* project.

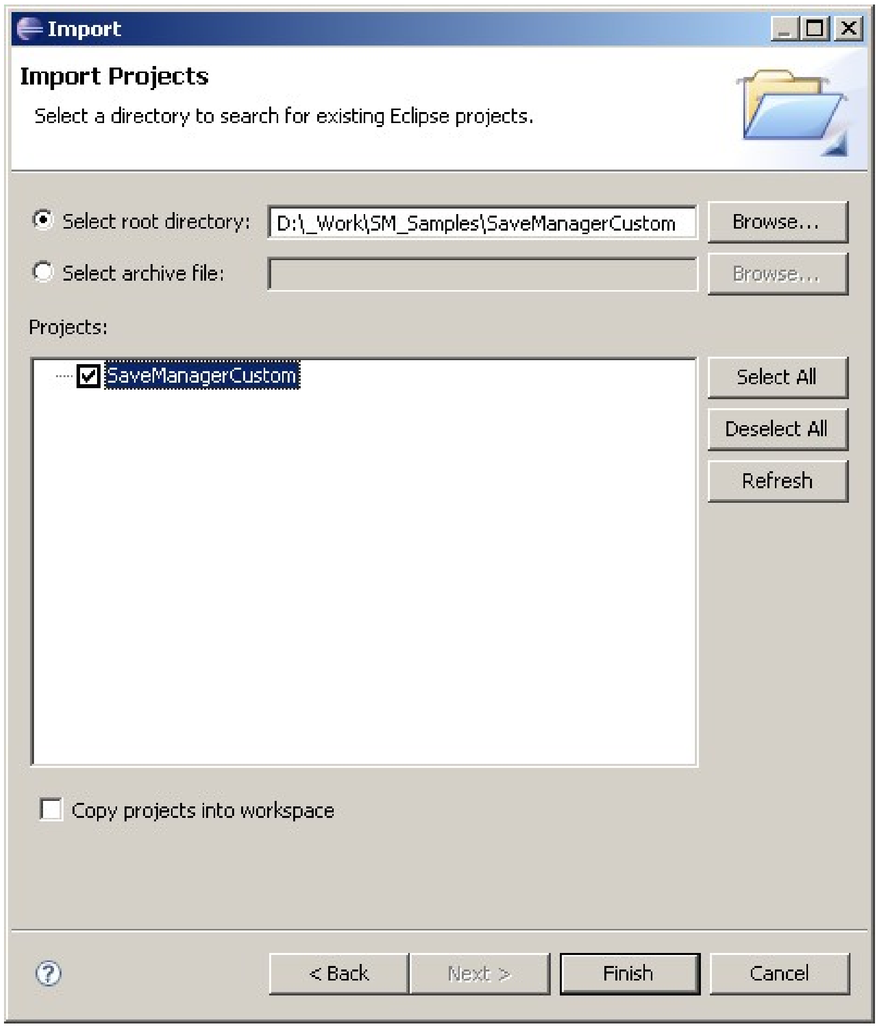
The sample project may be compiled using the Ant script in *buildCusto.xml* or by exporting the plug-in from Eclipse.

1. In Eclipse, pick **File >> Import**, then select **Existing Projects into Workspace** within the General folder.

2. Pick **Next** to continue.



3. Select the directory containing the extracted *projectSamples.zip* file, then pick **Finish**.



Select root directory: D:_Work\SM_Samples\SaveManagerCustom

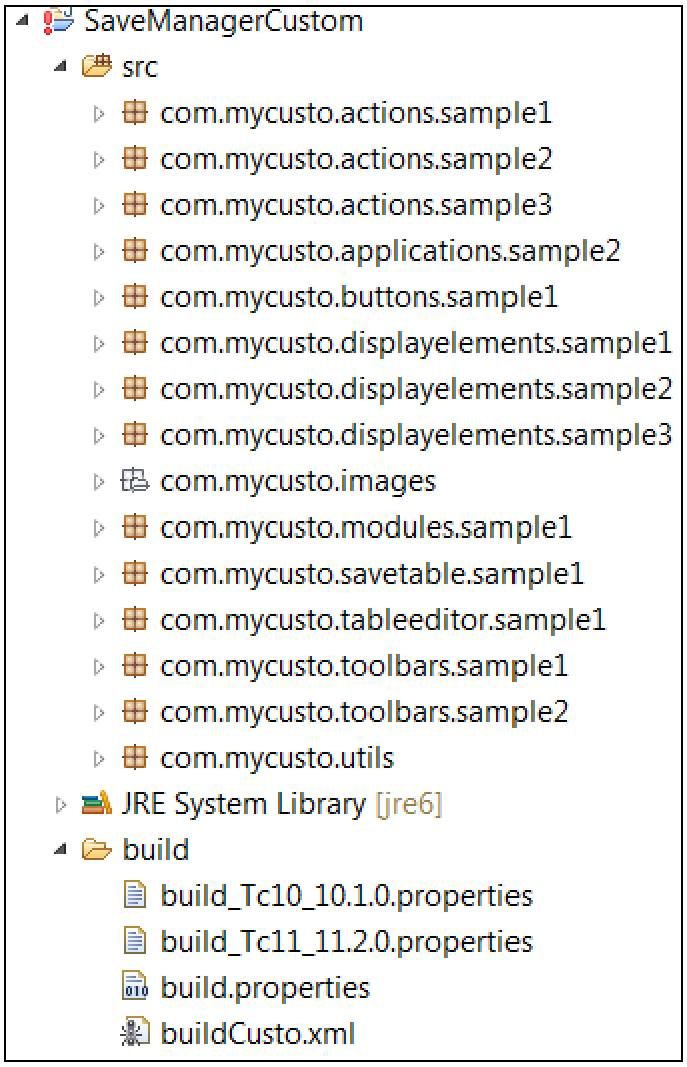
Select archive file:

Projects:

☑ SaveManagerCustom

☐ Copy projects into workspace

Result:



```
SaveManagerCustom
  src
    com.mycusto.actions.sample1
    com.mycusto.actions.sample2
    com.mycusto.actions.sample3
    com.mycusto.applications.sample2
    com.mycusto.buttons.sample1
    com.mycusto.displayelements.sample1
    com.mycusto.displayelements.sample2
    com.mycusto.displayelements.sample3
    com.mycusto.images
    com.mycusto.modules.sample1
    com.mycusto.savetable.sample1
    com.mycusto.tableeditor.sample1
    com.mycusto.toolbars.sample1
    com.mycusto.toolbars.sample2
    com.mycusto.utils
  JRE System Library [jre6]
  build
    build_Tc10_10.1.0.properties
    build_Tc11_11.2.0.properties
    build.properties
    buildCusto.xml
```

4. Change the properties of the *buildCusto.xml* file to successfully generate the SaveManagerCustom_3.0.0.jar file.

7.3.6.2 Create a New Display Element

Toolbars are designed to receive any type of JComponent, therefore, it is possible to create component types other than buttons. Three examples are provided:

1. Display the current date and time.

2. Display the number of selected elements in the table.

3. Indicate if a CATIA session has changed.

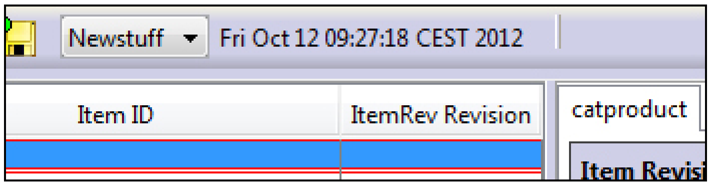
7.3.6.2.1 Display the Current Date and Time

The *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder, contains a class called **com.mycusto.displayelements.sample1.MyOwnDisplayElement**, which provides an example of integrating a new JLabel in the Save Manager. This JLabel indicates the current date and time by starting a timer to change the JLabel text and display the current date and time each second. This class is very small.

The *savemanager.xml* definition for this class, shown below, can be accessed within the *savemanager-displayelements-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder. Add this definition after the destination_folder DisplayElement.

```
<DisplayElement
    identifier="date_label"
    class="com.mycusto.displayelements.sample1.MyOwnDisplayElement"
    action="com.ebsolutions.uimanager.actions.SelectFolderAction"/>
```

The Save Manager now contains a new label next to the Newstuff box, and displays the current date and time.



When a display element such as a JLabel is redefined, the action attribute is not used because an action cannot be used with a JLabel. However, the action attribute must remain in the DisplayElement declaration within the xml file to avoid exception errors.

7.3.6.2.2 Display the Number of Selected Elements

The *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder contains a class called **com.mycusto.displayelements.sample2.MyOwnDisplayElement2** which provides an example of retrieving SaveModule information. This display element creates a JLabel to display the number of selected elements in the Save Table.

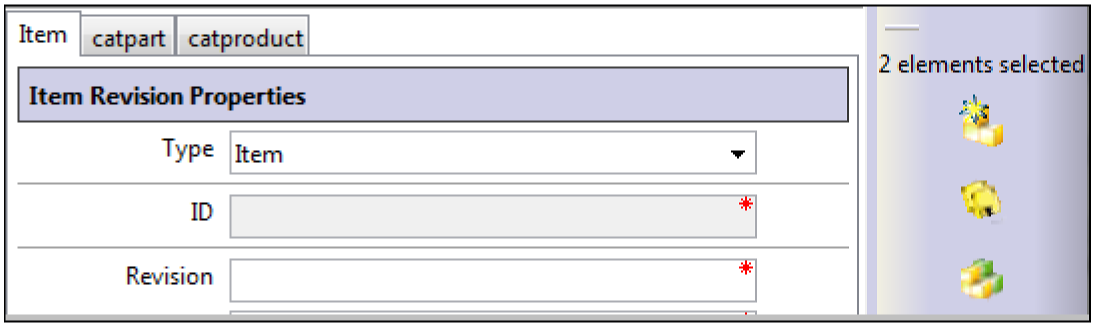
- Implement the ListSelectionListener interface.

- Redefine the valueChanged(ListSelectionEvent t) method.
- Add the class as a listener to the Table Selection with the `module.getTable().getSelectionModel().addListSelectionListener(this);` code.

The *savemanager.xml* definition for this class, shown below, can be accessed within the *savemanager-displayelements-sample2.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder. This sample must only be used in the Save *Module* Toolbar. Placing this sample into the Save *Manager* Toolbar causes a nullpointerException error because it attempts to get a module that has not been created at this stage of the button creation process. The number of selected elements is specific to the Save Module, and can therefore only be applied toward the Save Module.

```
<DisplayElement nameRef="Selected Elements" identifier="selected_elements"
  class="com.mycusto.displayelements.sample2.MyOwnDisplayElement2"
```

Each time the Save Table selection changes, it displays the number of lines selected.

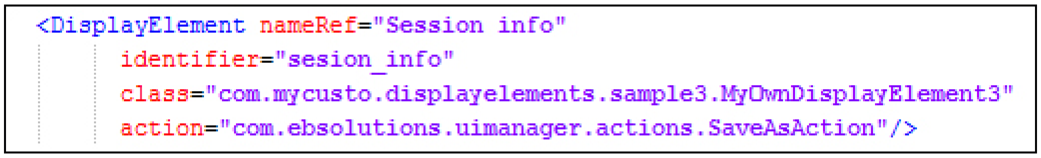


7.3.6.2.3 *Indicate if a CATIA Session Changed*

The *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder contains a class called **com.mycusto.displayelements.sample3.MyOwnDisplayElement3**, which provides an example of how to listen to a AbstractUIManagerSession. This DisplayElement creates a JLabel to indicate if the AbstractUIManagerSession has changed. When the Save Manager displays, this label indicates that the session has not changed, but if a part's property is changed and Applied, the label indicates the session changed. The object is notified in the sessionChanged method.

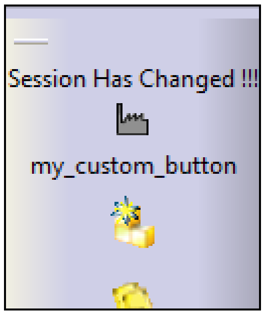
- Implement the AbstractUIManagerSessionListener Interface.
- Redefine the sessionChanged(AbstractUIManagerSessionEvent t) method.
- Add the component as a listener of the AbstractUIManagerSession.
- With the AbstractUIManagerSession object, use the addSessionListener(this); code

The *savemanager.xml* definition for this class, shown below, can be accessed within the *savemanager-displayelements-sample3.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.



```
<DisplayElement nameRef="Session info"
        identifier="sesion_info"
        class="com.mycusto.displayelements.sample3.MyOwnDisplayElement3"
        action="com.ebsolutions.uimanager.actions.SaveAsAction"/>
```

Each time the AbstractUIManagerSession changes, the label is notified and displays a message.



7.3.6.3 Create a New Action

The *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder contains three sample actions:

1. Select all lines in the Save Manager Table
2. Assign an Item Revision (CATIA data).
3. Set the Item Revision to the Item ID (CATIA data).

7.3.6.3.1 Select all lines in the Save Manager Table

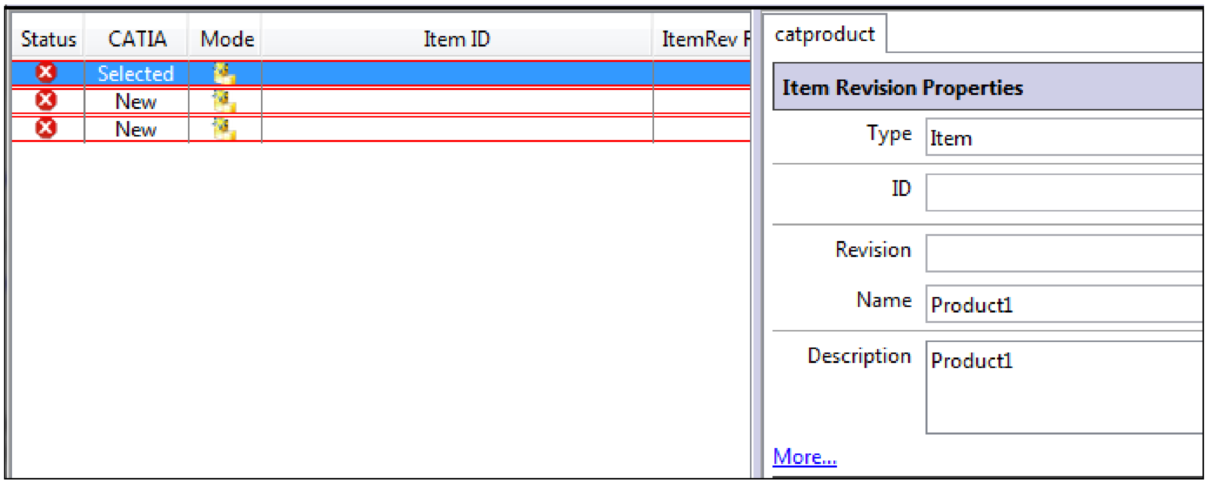
The com.mycusto.actions.sample1.MyOwnAction class selects all lines in the Save Manager Table.

- Retrieve the AbstractUIManagerApplication, then get the SaveModule.
- Get the CATIATable and its selection model.
- Select all the lines on the selection model.

The sample was tested with the custom button created earlier. The *savemanager.xml* declaration that applies the com.mycusto.actions.sample1.MyOwnAction action to the custom button is shown below, and can be accessed within the *savemanager-actions-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

```
<DisplayElement nameRef="My Button" identifier="my_custom_button" speed="10"
  image="/com/mycusto/images/mybutton.gif"
  class="com.mycusto.buttons.sample1.MyOwnButtonStyle"
  action="com.mycusto.actions.sample1.MyOwnAction" />
```

Result: When the *my_custom_button* is selected, all lines of the Save Table become highlighted.



Status	CATIA	Mode	Item ID	ItemRev F
✗	Selected			
✗	New			
✗	New			

catproduct

Item Revision Properties

Type	Item
ID	
Revision	
Name	Product1
Description	Product1

More...

7.3.6.3.2 *Assign an Item Revision (CATIA data)*

The **com.mycusto.actions.sample2.MyOwnAction2** class assigns the Item Revision Name to the Item ID of the *root part* of the CATIA assembly. This sample demonstrates how to access the CATIA Structure. Every AbstractUIManagerComponent can be accessed from the CATIASession and assigned a property.

The Item ID property has been set to an *editable* status with the following code. By default, this property is not editable.

```
Property itemIDProp =
(Property)comp.getComponentProperty(AbstractUIManagerComponent.ITEM
_ID);
itemIDProp.setEditable(true);
```

For editable properties, this step is not required and will only be necessary to set the property in the AbstractUIManagerComponent.

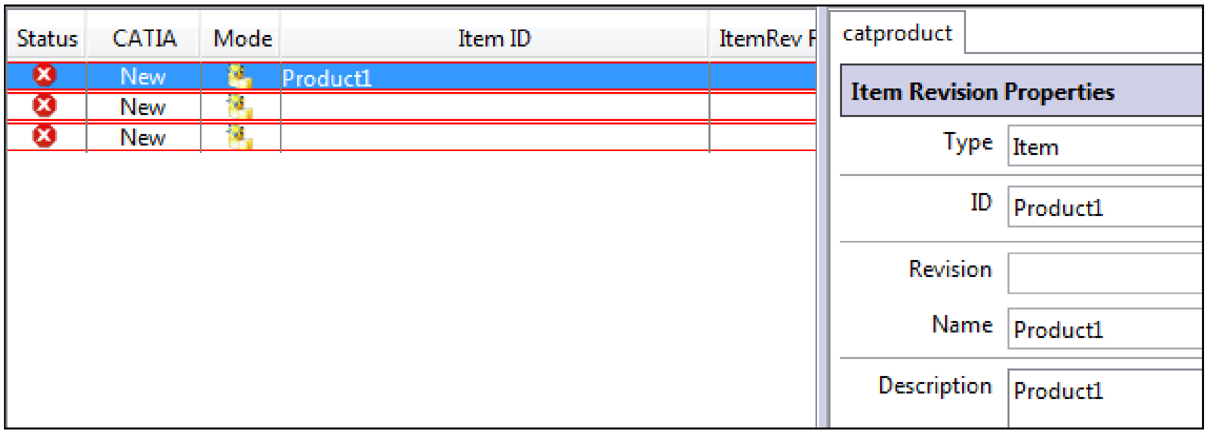
The sample was tested with the custom button created earlier. The *savemanager.xml* declaration that applies the MyOwnAction2 action to the custom button is shown below, and can be accessed within the *savemanager-actions-sample2.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

```
<DisplayElement nameRef="My Button" identifier="my_custom_button" speed="10"
  image="/com/mycusto/images/mybutton.gif"
  class="com.mycusto.buttons.sample1.MyOwnButtonStyle"
  action="com.mycusto.actions.sample2.MyOwnAction2" />
```

- The module.saveSelection() and module.updateSelection() calls are designed to keep the selection in the table, otherwise, the current table selection is lost after the action.

- The session.check() call is designed to update the table and parts status. If the property set causes a status error, the session.check() function disables the Save button. The check function causes every session listener to update. The SaveModule is listening to the session and updates the Table display as necessary.

Result: When the *my_custom_button* is selected, the root part Item ID is set to its Item Name.



Status	CATIA	Mode	Item ID	ItemRev F
⊗	New		Product1	
⊗	New			
⊗	New			

catproduct

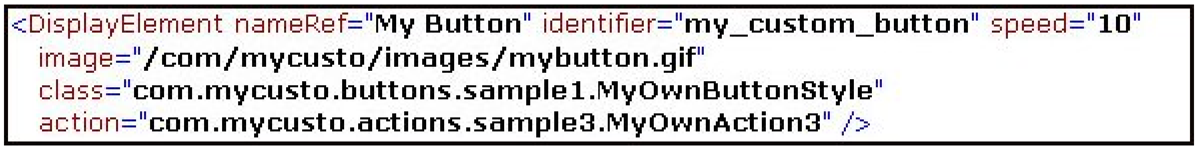
Item Revision Properties

Type	Item
ID	Product1
Revision	
Name	Product1
Description	Product1

7.3.6.3.3 Set the Item Revision to the Item ID (CATIA data)

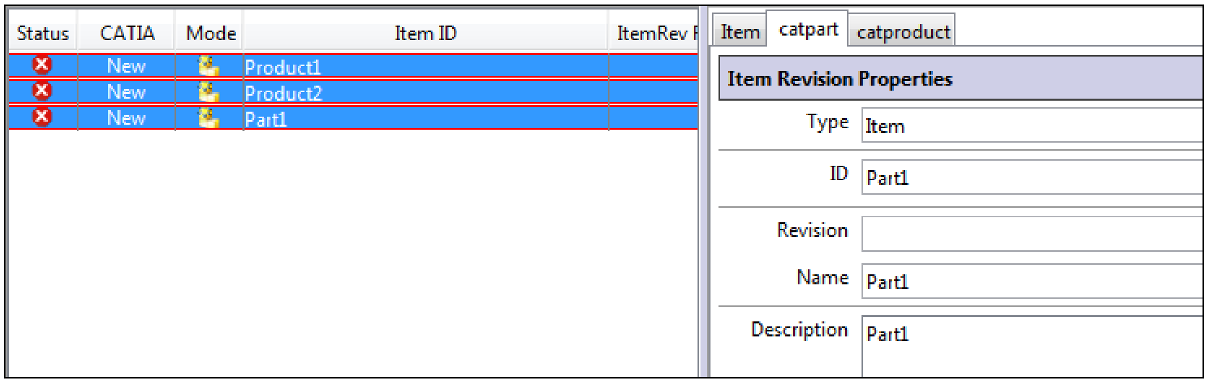
The **com.mycusto.actions.sample3.MyOwnAction3** class sets the Item Revision Name to the Item ID on all selected parts, instead of the Root Part only. This sample demonstrates how to access the table selected elements. The module.getSelectedElements() method returns every selected AbstractUIManager Component and InterfaceCATIAComponents that are cast into AbstractUIManagerComponents.

The sample was tested with the custom button created earlier. The *savemanager.xml* declaration that applies the com.mycusto.actions.sample3.MyOwnAction3 action to the custom button is shown below, and can be accessed within the *savemanager-actions-sample3.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.



```
<DisplayElement nameRef="My Button" identifier="my_custom_button" speed="10"
  image="/com/mycusto/images/mybutton.gif"
  class="com.mycusto.buttons.sample1.MyOwnButtonStyle"
  action="com.mycusto.actions.sample3.MyOwnAction3" />
```

Result: When the *my_custom_button* is selected, the Item ID of the selected parts are set to their Item Name.



Status	CATIA	Mode	Item ID	ItemRev I
❌	New		Product1	
❌	New		Product2	
❌	New		Part1	

Item Revision Properties

Type	Item
ID	Part1
Revision	
Name	Part1
Description	Part1

7.3.6.4 Create a New Application

The sample illustrates how to create an application that is using a:

- custom module
- custom module handler
- custom toolbar
- custom display element
- custom display action.

The sub-module handler,com.mycusto.applications.sample1.MyOwnAppliModuleHandler illustrates how to add tags in an xml module definition. The **com.mycusto.applications.sample2.MyOwnApplication** includes every redefinition possible.

Two display elements contain a custom action which displays a file chooser when a menu button is selected, permitting the user to pick .jpg files. For instance, to permit a selected picture to be rendered in a module. In this sample file, however, no action is defined when the file is selected.

The **com.mycusto.applications.sample2.MyOwnApplication2** application in the *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder illustrates how to change the display of the Save Manager. Once implemented, this application sample displays the Save

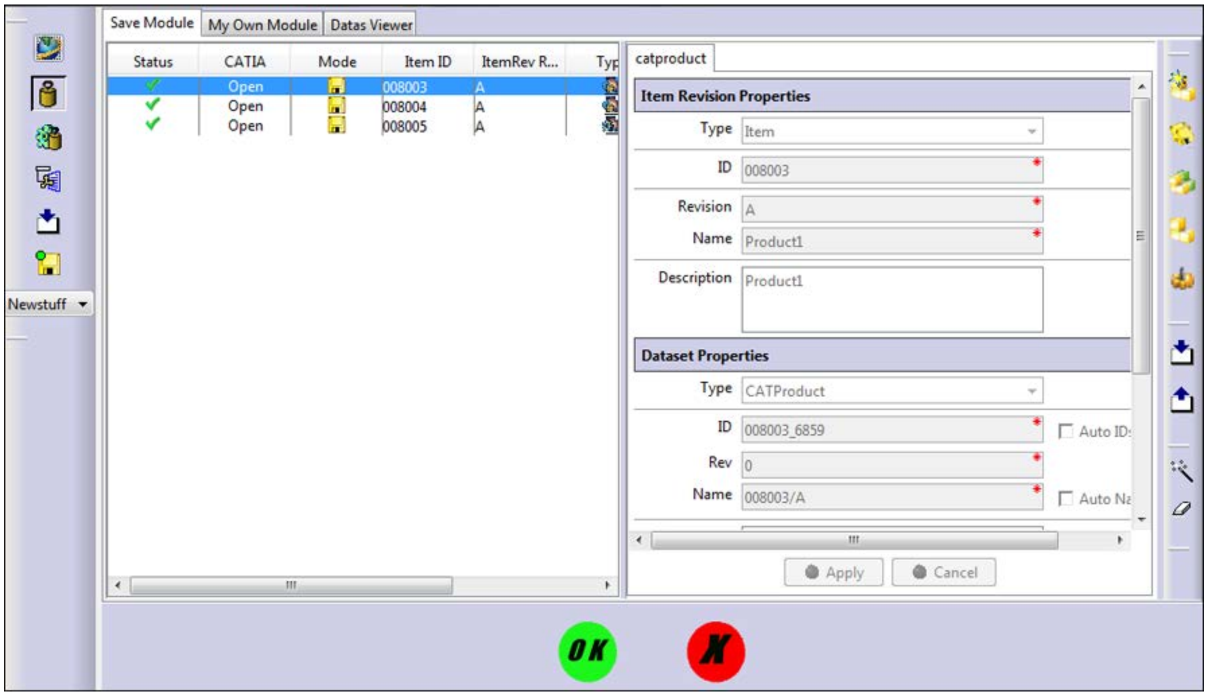
Manager toolbars on the left, the modules display inside a tabbed pane and the OK and Cancel buttons are redefined. This Application contains the following custom class declarations and three methods:

Custom Class Declarations/Methods	
<UIApplication>	com.mycusto.applications.sample2.MyOwnApplication2
<SaveManagerToolBar>	com.mycusto.applications.sample2.MyOwnToolBarStyle3
Module 1;	com.mycusto.applications.sample2.MyOwnAppliModule
Module 1 handler	com.mycusto.applications.sample2.MyOwnAppliModuleHandler
Module 2;	com.mycusto.applications.sample2.MyOwnModule
Module 2 handler	com.mycusto.applications.sample2.MyOwnModuleHandler
layoutApplication() method	called to render the Save Manager application. Every component can be accessed from within this method. Ensure it is determined how the components are displayed in the frame.
showApplication() method	called to set the Save Manager frame visible. This method is also an entry point to manage data before displaying the Save Manager and also after the Save Process has been confirmed.
getStatusBar() method	required to return a Status Bar to the Save Module. Redefining this method is not required if the Save Module is not used in your *savemanager.xml* declaration.

The *savemanager.xml* definition for this sample can be accessed within the *savemanager-applications-sample2.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

This sample contains actionlisteners for the OK and Cancel buttons to define the Application status each time the buttons are selected. The release() method must also be called to dispose the frame when the buttons are selected because the super.showApplication(frame); method is waiting for the release() command to dispose the frame.

These activities can be customized based on the release of the frame or the status of the OK and Cancel buttons, depending on CATIA data, for example.



Status	CATIA	Mode	Item ID	ItemRev R...	Typ
✓	Open		008003	A	
✓	Open		008004	A	
✓	Open		008005	A	

Item Revision Properties

Type: Item
ID: 008003
Revision: A
Name: Product1
Description: Product1

Dataset Properties

Type: CATProduct
ID: 008003_6859 ☐ Auto ID:
Rev: 0
Name: 008003/A ☐ Auto Na

7.3.6.5 Create a New Button

Redefine the Toolbar

Apply the Button Feature

A button is a DisplayElement tag in the *savemanager.xml* file, whose extendable class is the AbstractDisplayElement class.

- Extending the AbstractDisplayElement class automatically provides the xml tags (described in the The Data Accessor Methods section).

- The createComponent() method must be redefined. Use the createComponent() method to create your button as a JComponent by applying the button style desired and add it to the end of the function.

- The getID() method must be redefined. The getID() method is designed to return a unique identifier for this DisplayElement. The *name* XML attribute value is used for the ID in the example below.

7.3.6.5.1 com.mycusto.buttons.sample1.MyOwnButtonStyle Class Example

The **com.mycusto.buttons.sample1.MyOwnButtonStyle** class in the *projectSamples.zip* file, in the *TcIC_DIR\custom_client\SaveManager* folder, describes a button whose icon rotates each time the mouse hovers on it. The **com.mycusto.utils.Rotator** class, provided in the sample project, calculates the icon rotations.

The Rotator class is a sample class used to demonstrate that it is possible to use customized utilities and include them within your package.

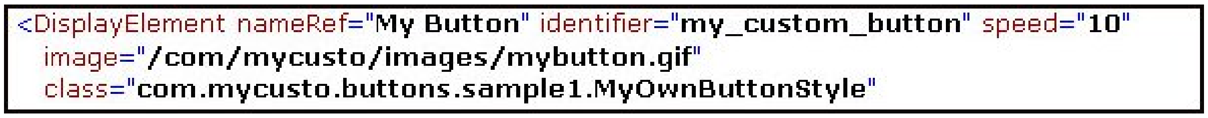
The com.mycusto.buttons.sample1.MyOwnButtonStyle class:

- changes the appearance of the SaveManager button by using new xml attributes from the *savemanager.xml* file and used in the code.

- contains speed attributes added to the xml file, which passes through the initialize(Attributes attributes) method redefinition and is applied to the timer rotating the button icon.

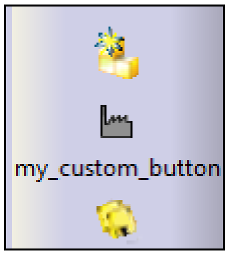
- Listeners have been added on the button created in the createComponent() method to start/stop the rotation timer.

The *savemanager.xml* definition for this button, shown below, can be accessed within the *savemanager-buttons-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.



```
<DisplayElement nameRef="My Button" identifier="my_custom_button" speed="10"
  image="/com/mycusto/images/mybutton.gif"
  class="com.mycusto.buttons.sample1.MyOwnButtonStyle"
```

The Save Manager contains a new button named *my_custom_button* and rotates when the mouse hovers on it.



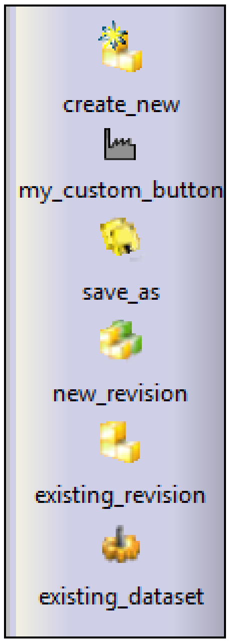
7.3.6.5.2 *Redefine the Toolbar*

The toolbar must also be defined when a button has been redefined in the Save Manager. By default, the SaveManagerToolbar is designed to control its elements. The button implementation will be manipulated by the toolbar to fit in size. In addition, every implemented button is not opaque and the text of the button is mandated by the toolbar. If a specific font is applied to the button, it will not display because the text is separated into a JLabel by the toolbar.

> If necessary, redefine the complete Toolbar to accommodate specific needs.

7.3.6.5.3 *Apply the Button Feature*

Replace the `class` attribute value of every DisplayElement definition in the *savemanager.xml* file to apply this button feature to every display element in the Save Manager. For example, if *MyOwnButtonStyle* is applied to every button in the *savemanager.xml* file, it will display as shown.



7.3.6.6 Create a New Module

Notes

Important Warning

Schema and Location

Result

The projectSamples.zip file in the *TcIC_DIR\custom_client\SaveManager* folder contains a Module implementation demonstrating how to create a module. It is not a fully functional module, its intention is to illustrate module creation, determine differences between modules, and demonstrate how modules listen to their environment.

The **com.mycusto.modules.sample1.MyOwnModule** module in the *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder contains a **com.mycusto.modules.sample1.MyTree** class which creates a Teamcenter tree corresponding to the item revisions to be Saved. This tree has the same look & feel as the My Teamcenter tree and can be useful for viewing the item revision inside the Save Manager, rather than in Teamcenter. This tree implementation constructs an object tree from the com.ebsolutions.uimanager.AbstractUIManagerApplication class. By default, the root node is the **Home** folder and children are the item revisions corresponding to the CATIA parts. New parts display with a question mark.

The module also contains the **com.mycusto.modules.sample1.EmptyComponent** class which extends the IMANComponent type to render an empty element in the Tree. Because this tree uses an TcTreeCellRenderer, it is useful to give a toString() implementation for an empty element.

7.3.6.6.1 Notes

- This module extends an AbstractModule because this module does not contain Save Module behaviors or displays. If it did, it would have extended the AbstractSaveModule class.
- Three interfaces are implemented: AbstractUIManagerSessionListener, ApplicationListener and ListSelectionListener to ensure the module listens to the AbstractUIManagerSession, the Application and the Save Module Table.
- Methods added for the listeners are:
    - The *public void sessionChanged(final AbstractUIManagerSessionEvent ev)* method is invoked when the AbstractUIManagerSession changes.
    - The *public void applicationChanged(ApplicationEvent t)* method is invoked when the application changes.

    - The *public void valueChanged(ListSelectionEvent e)* method is invoked each time the Save Module Table selection changes.

- This module may or may not be combined with the Save Module in the savemanager.xml file. If the Save Module is not involved, the listening methods for the Save Module are not invoked.

- This module can communicate information to the Save Module. For example, when an item revision is selected in the tree, the `treeselectionlistener` transmits information to the Save Module.

- The Module Handler must also be redefined because the existing Save Module Handler is not compatible with this module. The **com.mycusto.modules.sample1.MyOwnModuleHandler** extends the AbstractModuleHandler class. To avoid using xml tags, it does not contain redefinitions.

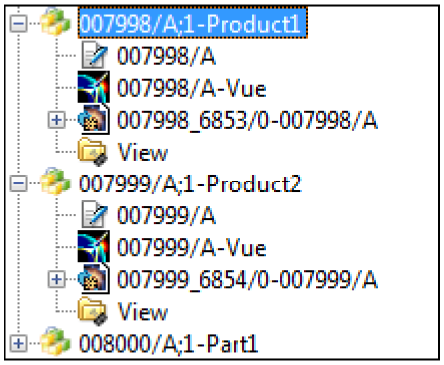
- When the application contains more than one module, each module accesses other modules by retrieving the application using the getModules() application method.

7.3.6.6.2 Important Warning

Using the Datas Viewer module alone in the *savemanager.xml* definition causes Save errors because the Integration interface Save process requires a Save Module.

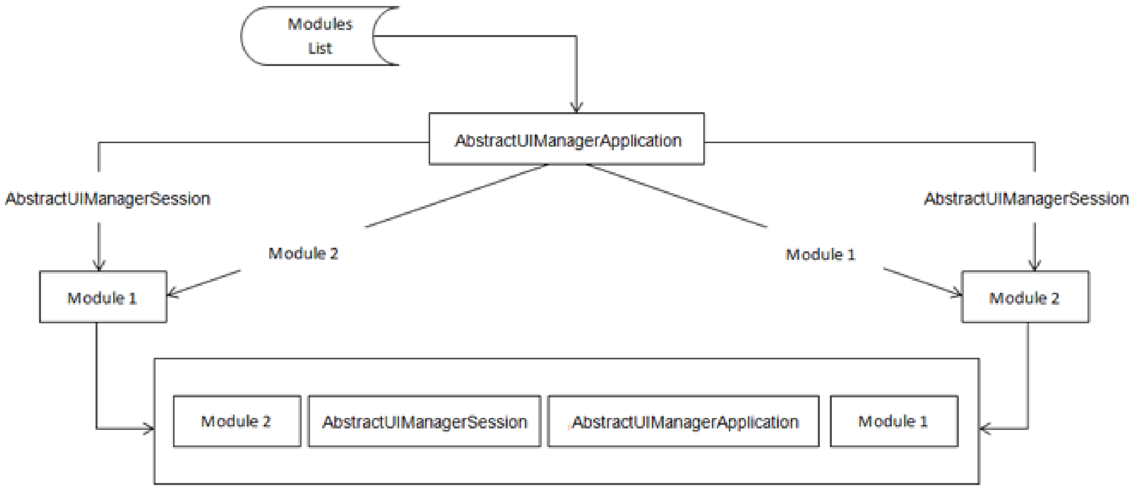
Create a standalone module by extending the com.ebsolutions.uimanager.modules.SaveModule class to fit the Save process needs.

NOTE:
The Filter button displayed on the bottom right corner of the Save Manager is declared in the createComponent() method of the Save Module implementation. This capability is not implemented when using a standalone module.



7.3.6.6.3 *Schema and Location*

Refer to the schema below for additional information.

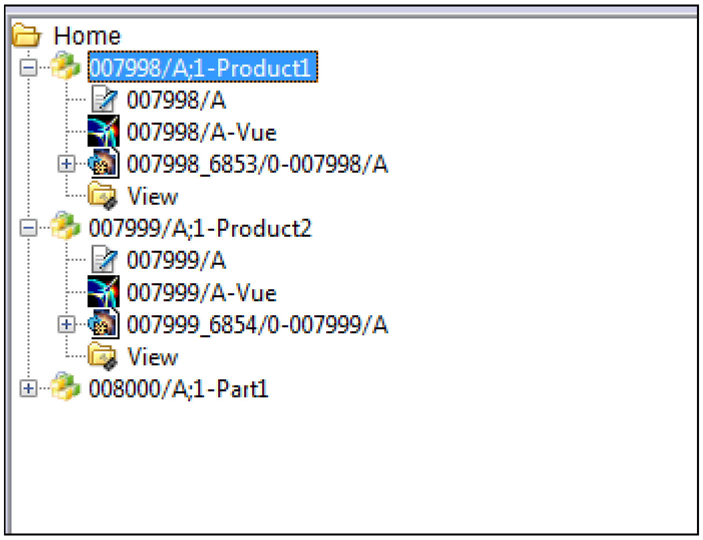


The *savemanager.xml* definition to implement the com.mycusto.modules.sample1.MyOwnModule module can be accessed within the *savemanager-modules-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

When adding toolbar definitions, use the commented code in the sample as a guide. The getToolbars() method retrieves all the module toolbars and the getDisplay() method gets their display. In the example provided, no toolbar is defined for the Datas Viewer module.

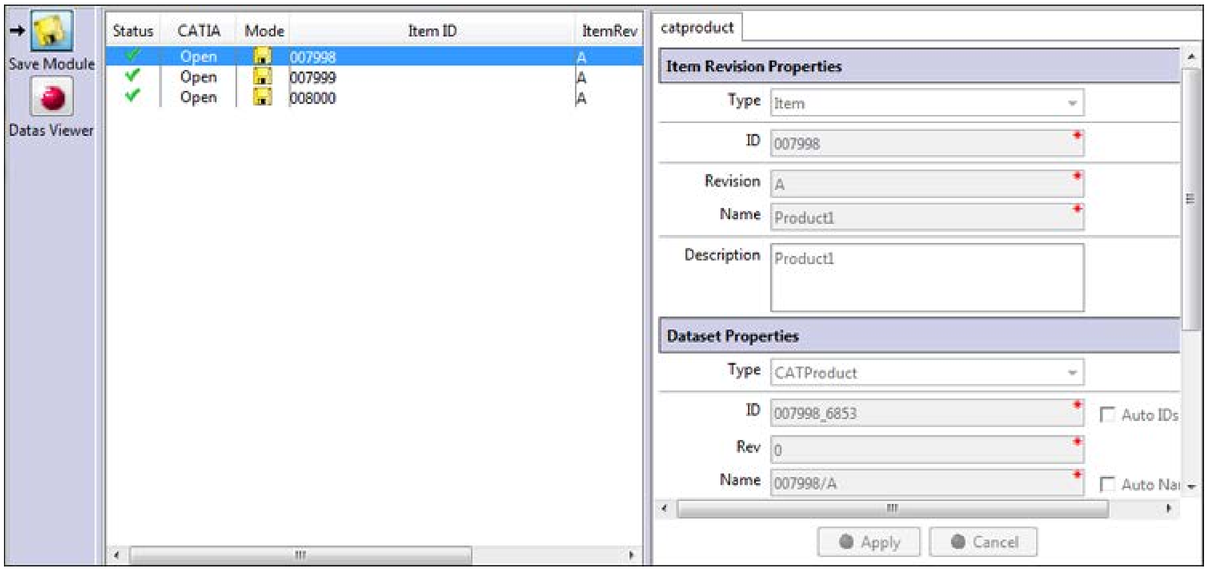
7.3.6.6.4 Result

Result of the com.mycusto.modules.sample1.MyOwnModule module:



Home
- 007998/A;1-Product1
  - 007998/A
  - 007998/A-Vue
  - 007998_6853/0-007998/A
  - View
- 007999/A;1-Product2
  - 007999/A
  - 007999/A-Vue
  - 007999_6854/0-007999/A
  - View
- 008000/A;1-Part1

The Save Module is used in parallel with the Datas Viewer module, i.e, each line selection also selects the corresponding item revision in the Data Viewers module.



Status	CATIA	Mode	Item ID	ItemRev
✓	Open		007998	A
✓	Open		007999	A
✓	Open		008000	A

catproduct

Item Revision Properties

Type	Item
ID	007998
Revision	A
Name	Product1
Description	Product1

Dataset Properties

Type	CATProduct	
ID	007998_6853	☐ Auto IDs
Rev	0	
Name	007998/A	☐ Auto Nai

7.3.6.7 Create a New Toolbar Style

The *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder contains two sample toolbar redefinitions:

1. Define Button Fonts
2. Toolbar Display

7.3.6.7.1 Define Button Fonts

Creating a new toolbar implementation provides more accessibility to its contained elements. By default, the SaveManagerToolBar class drives its elements and prevents some display customizations.

The **com.mycusto.toolbars.sample1.MyOwnToolBarStyle** toolbar redefinition works in conjunction with the com.mycusto.toolbars.sample1.MyOwnToolBarStyleButton button redefinition. This sample demonstrates a toolbar which uses extra xml parameters to define the text font of the button. By redefining the button, the text can be changed and displayed as needed.

- The `font size`, `fontName`, `fontStyle` and `direction` attributes have been added to the toolbar definition.
- The initialize(Attributes attributes) method has been redefined in the com.mycusto.toolbars.sample1.MyOwnToolBarStyle class to manage the new settings.
- The defined font has been set, in the createComponent() method, to every display element and added to the toolbar.
- The `direction` attribute indicates the horizontal or vertical direction of the toolbar and the display element disposition changes depending on this attribute.
- A gradient color is used in this sample to display the toolbar to keep the same default Save Manager toolbar display.

The xml definition defines this toolbar style for the Save Manager toolbar and for the Save Module toolbar. The difference between the toolbars is the font of the display elements.

This customization can be useful in modifying fonts *on the fly* within the xml file.

The *savemanager.xml* definition for this toolbar style is shown below and can be accessed within the *savemanager-toolbars-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

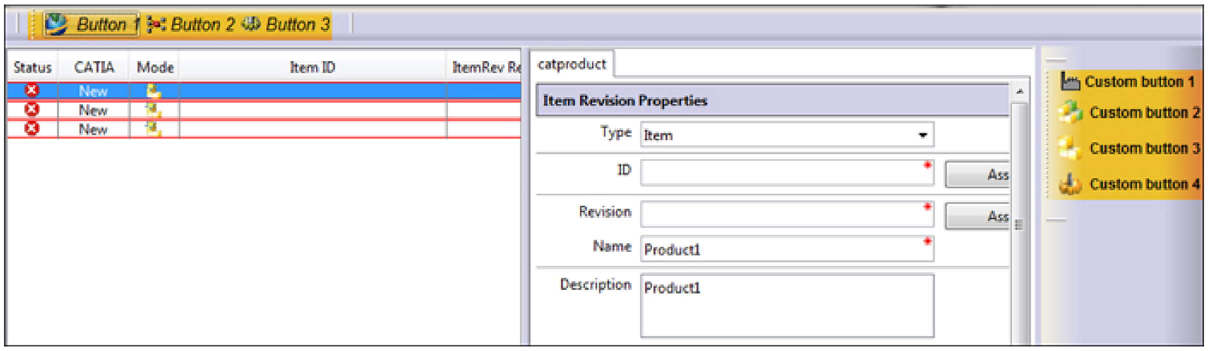
The Save Manager Toolbar xml definition:

```
<SaveManagerToolBar nameRef="toolbar1" identifier="tb1" fontSize="15" fontName="Dialog"
  fontStyle="Font.ITALIC" class="com.mycusto.toolbars.sample1.MyOwnToolBarStyle">
```

The Module Toolbar xml definition :

```
<ToolBar nameRef="SAVE_TOOL_BAR" identifier="saveModuleToolBar" fontSize="12"
  fontName="Dialog" fontStyle="Font.BOLD" direction="vertical"
  class="com.mycusto.toolbars.sample1.MyOwnToolBarStyle">
```

Result:



7.3.6.7.2 *Toolbar Display*

The **com.mycusto.toolbars.sample2.MyOwnToolBarStyle2** toolbar redefinition works in conjunction with the com.mycusto.toolbars.sample2.MyOwnToolBarStyleButton2 button redefinition. This sample demonstrates a toolbar with a different display and may be used to provide customization ideas. The sample toolbar is a **JMenuBar**.

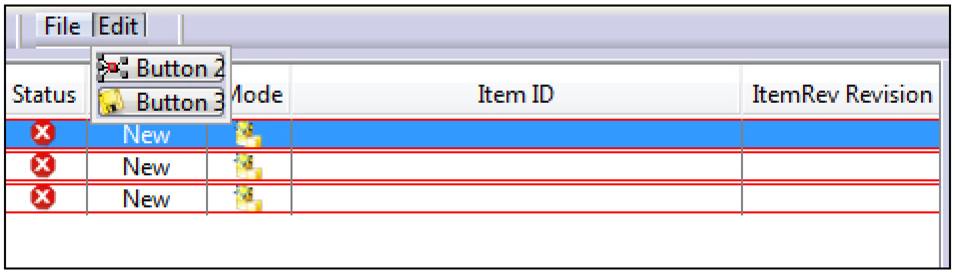
- The public void initialize(Attributes attributes) method in the MyOwnToolBarStyleButton2 button redefinition class has been redefined to manage a new xml attribute called `category`.
- The `category` attribute indicates which toolbar contains the button menu.
- One button has been added within the **File** menu and two buttons have been added within the **Edit** menu using the category attribute.

- The MyOwnToolBarStyle2 toolbar class retrieves every Display element and calls the getCategory() method defined in the MyToolBarStyleButton2 class for each of them. This indicates to the toolbar the number of menus required, then adds the corresonding elements using the getDisplay() method.

The MyOwnToolBarStyle2 class is only valid with MyOwnToolBarStyleButton2 Display elements. A cast is explicitly performed in the toolbar, required to get the getCategory() method.

The *savemanager.xml* definition can be accessed within the *savemanager-toolbars-sample2.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder.

Result :



7.3.6.8 Manage CATIAdata Before and After the Save Manager Displays

The Save Manager element controlling the Save Manager JFrame is the Application. The showApplication() method displays the application. Accessing CATIA data before and after the Save Manager frame displays is possible when this method is redefined within an Application implementation.

The method public synchronized in the showApplication(Frame frame) is redefined in the com.mycusto.applications.sample2.MyOwnApplication2 Application. The entry point before the Save Manager display is just before the call to super.showApplication(frame) and the entry point after the Save Manager display is just after this call.

Refer to the following example to include the MyOwnApplication2 showApplication(Frame) method in order to add a specific action to the CATIA data before and after the Save Manager displays.

```
public synchronized int showApplication(Frame frame)
        {
                CATIASession catiaSession =
(CATIASession)getSaveManagerSession().getCatiaSession();
                for(int a - 0 ; <
catiaSession.getComponents().size() ; a++)
```

```
                {
                        CATIAComponent cmp -
catiaSession.getComponent(a);
                        Property prop -
(Property)cmp.getComponentProperty(CATIAComponent.ITEM_NAME);
                        prop.setEditable(true);
                       cmp.setProperty(CATIAComponent.ITEM_NAME,"Pa
rtNumber_"+a);
                        IMANComponentItemRevision itemRev -
MyUtils.getItemRevision(catiaSession.getIMANSession(),cmp);
                        if(itemRev -- null)
                        {
                                 System.out.println("CATIA Document
number "+a +" is existing in Teamcenter");
                        }
                        else
                        {
                                 System.out.println("CATIA Document
number "+a +" is not existing in Teamcenter");
                        }
       }
       super.showApplication(frame);

       /*At this step the Save Manager is disposed but you can
still access CATIA datas*/ return getAppliStatus(); }
```

Notice a new value to the item_name property of each document. When the document exists in Teamcenter, the properties are not editable. Use the setEditable(true) method to allow it to be edited.

The MyUtils.getItemRevision(InterfaceCATIAcomponent) method, called in a commented source in this sample, is useful because it retrieves the Item Revision corresponding to the CATIAComponent. Refer to the com.mycusto.utils.MyUtils class for other useful methods used between the CATIA document and Teamcenter objects.

7.3.6.9 Change the Table Display

List of CATIA Property Types

The Save Table is a specific Save Module component. The **com.ebsolutions.uimanager.modules.CATIATable** class is the implementation delivered with the Save Manager toolkit.

The com.ebsolutions.uimanager.modules.CATIATable extends the com.ebsolutions.uiapplication.ui.UIApplicationTable class. The AbstractSaveModule createDefaultTable() method waits for a UIApplicationTable type. Create a class that extends the com.ebsolutions.uiapplication.ui.UIApplicationTable class (or the com.ebsolutions.uimanager.modules.CATIATable, also a UIApplicationTable) to create your own table implementation.

The **com.mycusto.savetable.sample1.MyOwnTable** class in the *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder redefines a Table implementation by redefining the createDefaultTable() method to customize the Save Table.

Customize the Save Table by creating a new module to extend the SaveModule. A sample class called **com.mycusto.savetable.sample1.MyOwnModule3**, is provided in the *projectSamples.zip* file in the *TcIC_DIR\custom_client\SaveManager* folder. This sample class redefines the CreateDefaultTable() method, com.mycusto.savetable.sample1.MyOwnTable. Refer to the following code:

```
protected UIApplicationTable createDefaultTable()
{
MyOwnTable table = new MyOwnTable(this);
Font f - new Font("Dialog", Font.BOLD, 15);
table.getTableHeader().setFont(f);

return table;
    }
```

The implemented Table MyOwnTable extends the UIApplicationTable class and provides the ability to reorder and filter. The MyOwnModule3 module redefines a table implementation, but the data model has not been redefined allowing the implemented MyOwnTable to interpret the CATIA data model the same as the original Save Module table interprets the CATIA data model.

The **createDefaultTable()** method is the entry point to change the Save Table. Use the MyOwnTable implementation to define new renderers, for example. Two custom CellRenderer implementations are provided within the sample file:

- com.mycusto.savetable.sample1.MyOwnStringCellRenderer
- com.mycusto.savetable.sample1.MyOwnIntegerCellRenderer.

Icon customizations are also provided in the two renderers and are accessible using the **com.mycusto.utils.MyUtils.getIcon(String)** static method.

Ensure icons are loaded only one time. If icons are created manually in the renderer, they load each time a cell displays causing the Save Manager to display slowly.

These renderers manage the cell display in the table depending on data type. Refer to the code below when customizing display of the call, depending on its CATIAComponent property the column displays.

```
if (model.getColumnID(index).equals(CATIAComponent.STATUS))
{
processStatus(((Integer)value).intValue());
}
```

The two renderers provided manage all properties containing a String and Integer type. The Boolean and List type renderes used in this table sample are used in the Save Manager implementation. The *out of the box* renderers are :

- com.ebsolutions.uimanager.modules.StringCellRenderer
- com.ebsolutions.uimanager.modules.IntegerCellRenderer
- com.ebsolutions.uimanager.modules.TypeListCellRenderer
- com.ebsolutions.uimanager.modules.BooleanCellRenderer

When implementing a new cell renderer, do not extend the classes listed above in order to avoid problems with the icons.

The 2 renderers provided have similar codes as the classes above and can be used to customize columns as desired. The provided renderers do not extend the above classes, but extend the basic Save Manager renderers.

- com.ebsolutions.uiapplication.ui.DefaultStringSaveManagerCellRenderer
- com.ebsolutions.uiapplication.ui.DefaultSaveManagerCellRenderer
- com.ebsolutions.uiapplication.ui.DefaultIntegerSaveManagerCellRenderer
- com.ebsolutions.uiapplication.ui.DefaultDoubleSaveManagerCellRenderer
- com.ebsolutions.uiapplication.ui.DefaultBooleanSaveManagerCellRenderer

Using the renderers provided will avoid concrete implementation constraints and can be completely customized. Note that every column cell rendering must then be subsequently redefined.

7.3.6.9.1 List of CATIA property types

Property name	Property type
`link_type`	Type
`file_name`	String
`former_file_name`	String
`document_type`	String
`modifiable_item`	Integer
`modifiable_revision`	Integer
`modifiable_dataset`	Integer
`modifiable_bvr`	Integer
`co_status`	Integer

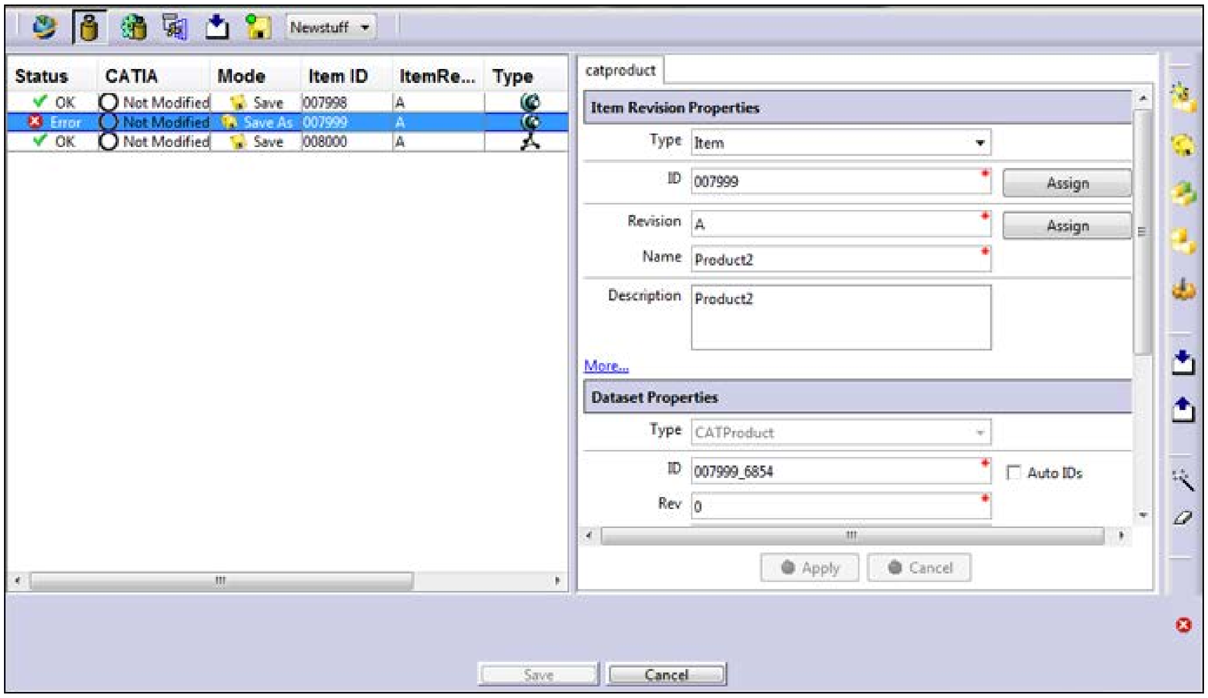
Property name	Property type
`catia_status`	Integer
`save_mode`	Integer
`status`	Integer

Extending the classes located in the com.ebsolutions.uimanager.modules package may cause graphics problems if every column is not redefined. The existing renderers will fail to find icons when using the super.getTableCellRendererComponent(...) method.

These two renderers also redefine the color of the error lines to blue. The createDefaultTable() method redefinition changes the font of the table header.

A module handler can be redefined with the addition of xml tags to change the table display. For example, a module handler can be created to retrieve a *tableHeaderSize* attribute, pass it to the module and apply it to the table header in the createDefaultTable() method.

The sample can be found within the *savemanager-savetable-sample1.xml* file in the *TcIC_DIR\custom_client\SaveManager* folder. The screenshot below is the result of the Save Manager interface using this module and save table implementation:



Status	CATIA	Mode	Item ID	ItemRe...	Type
OK	Not Modified	Save	007998	A	
Error	Not Modified	Save As	007999	A	
OK	Not Modified	Save	008000	A	

Type: Item
ID: 007999
Revision: A
Name: Product2
Description: Product2

Type: CATProduct
ID: 007999_6854
Rev: 0

7.3.7 Save Manager API Java Documentation

The Save Manager API Java Documentation is divided into two sections. Refer to the online documentation for access to these files.

File	Description
uimanager package	The Cat2uimanager.jar file under *%TCICV5_DIR%\racless* is the package distribution that contains all the necessary classes used to implement your own objects. Use the com.ebsolutions.uiapplication package to create a new Save

File	Description
uiapplication package	Manager interface, with specific, customized behavior. The Teamcenter implementation of the Save Manager contains complete data management and data representations redefined in the .jar file (in the InterfaceCATIASession and InterfaceCATIAcomponent interfaces) to fit Teamcenter metadata requirements. The implementation also drives the Save Manager data with the Teamcenter Server to synchronize visuals and metadata. Use the com.ebsolutions.uimanager package in your project for Teamcenter specific customizations, rather than creating a new implementation. This package contains implementations to manipulate Teamcenter data, and should be used as a source, rather than implementing a complete, new specific implementation. Refer to the packages section for descriptions.

7.4 User Exit Class

7.4.1 Entry Point Before Local Save

7.4.1.1 Overview

The **CV5ICAAEntryPoints::entryPointBeforeLocalSave** function is called before saving CATIA documents in the staging directory during Save processes. The following CATIA documents are affected by this entry point:

- CATProduct
- CATPart
- CATDrawing
- CATAnalysis
- CATProcess
- CATShape

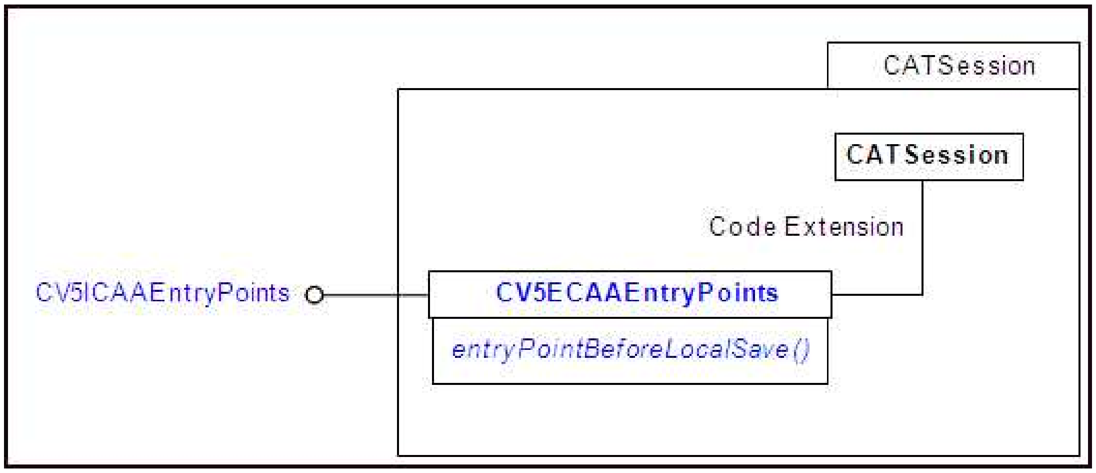
The CV5ICAAEntryPoints::entryPointBeforeLocalSave function can be customized using CATScript with *%TCICV5_DIR%/custom_CATScript/EntryPointBeforeLocalSave.CATScript* or CAA APIs, by re-implementing the CV5ECAAEntryPoints::entryPointBeforeLocalSave.

7.4.1.2 Customize Using CAA APIs

Perform the following steps to customize the *CV5ICAAEntryPoints::entryPointBeforeLocalSave* function using CAA APIs. The source code has been provided in the *%TCICV5_DIR%/custom_CAA/EntryPoints* folder.

Refer to the Modify CATIA Documents to modify CATIA documents in the entryPointBeforeLocalSave function.

1. Modify the implementation of the **entryPointBeforeLocalSave** of the CV5ECAAEntryPoints class function. This class is an implementation of the CV5ICAAEntryPoints interface.

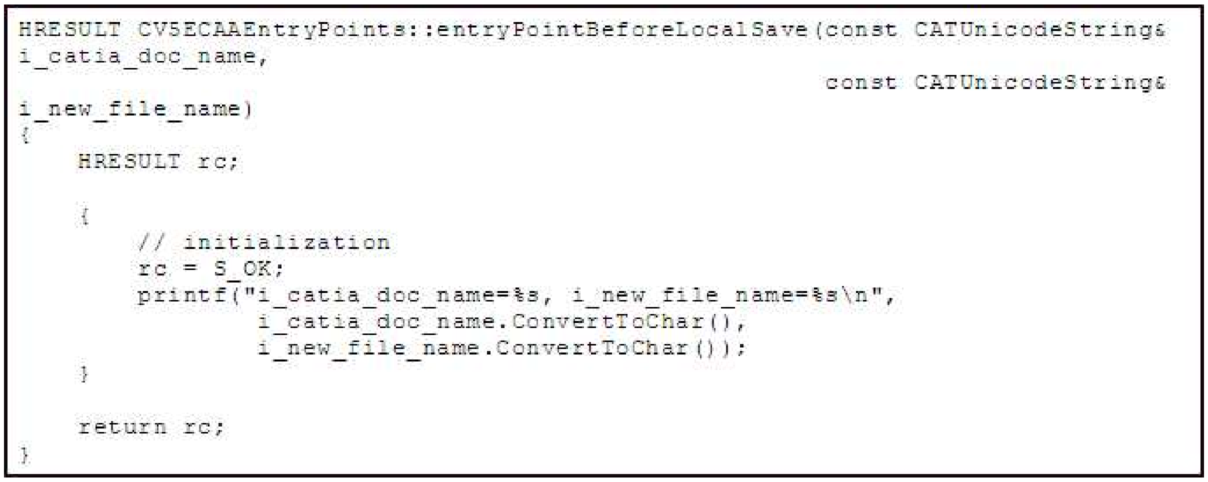


The source code of the entryPointBeforeLocalSave function has been provided in the *%TCICV5_DIR%/ custom_CAA/EntryPoints/CV5CAAEntryPoints/CV5CAAEntryPointsImpl.m/src/CV5ECAAEn* *tryPoints.cpp* folder.

This function contains two CATUnicodeString parameters:

- `i_catia_doc_name (I)` : storage name of the document in the CATIA session before save (full file name with path).
- `i_new_file_name (I)` : new document full file name after save in staging directory.

Refer to the example below which prints the `i_catia_doc_name` and `i_new_file_name` in the standard output.



```
HRESULT CV5ECAAEntryPoints::entryPointBeforeLocalSave(const CATUnicodeString&
i_catia_doc_name,
                                                      const CATUnicodeString&
i_new_file_name)
{
    HRESULT rc;

    {
        // initialization
        rc = S_OK;
        printf("i_catia_doc_name=%s, i_new_file_name=%s\n",
                i_catia_doc_name.ConvertToChar(),
                i_new_file_name.ConvertToChar());
    }

    return rc;
}
```

2. Execute *%TCICV5_DIR%/custom_CAA/EntryPoints/Custom_CAA_EntryPoints_compilation.vbs* (Windows) or *$TCICV5_DIR/ custom_CAA/EntryPoints/Custom_CAA_EntryPoints_compilation* (UNIX) to compile the CV5CAAEntryPoints framework.

   CAA libraries are created for the CV5CAAEntryPointsInterfaces.m and CV5CAAEntryPointsImpl.m modules. When CAA compilation finishes, old CAA libraries for the CV5CAAEntryPointsInterfaces.m and CV5CAAEntryPointsImpl.m modules are automatically updated in *%TCICV5_DIR%/bin/CAA/{CATIA_release}/{OS}*, where {CATIA_release} is V5R18, V5R19 or V5R20 and {OS} is intel_a (Windows 32-bit), win_b64 (Windows 64-bit), solaris_a (Solaris) or aix_a (AIX).

7.4.1.3 Customize Using CATScript

Perform the following steps to customize the CV5ICAAEntryPoints::entryPointBeforeLocalSave function using CATScript.

Refer to Modify CATIA Documents to modify CATIA documents in the CATMain procedure.

3. Create a file named *EntryPointBeforeLocalSave.CATScript* in the *%TCICV5_DIR%/custom_CATScript* folder.

4. Create a procedure within the EntryPointBeforeLocalSave.CATScript file, named *CATMain* with two parameters of type CATBSTR:

   - `i_catia_doc_name (I)`: storage name of the document in the CATIA session before save (full file name with path)

   - `i_new_file_name (I)`: new document full file name after save in staging directory

The example below prints the `i_catia_doc_name` and `i_new_file_name` in the standard output.

```
''
' This function display parameters i_catia_doc_name and
' i_new_file_name in a message box.
'
Sub CATMain(i_catia_doc_name, i_new_file_name)

    MsgBox "i_catia_doc_name=" & i_catia_doc_name & ", i_new_file_name=" &
i_new_file_name

End Sub
```

7.4.1.4 Modify CATIA Documents

Perform the following steps to modify CATIA documents.

1. Ensure the document to modify is opened in a CATIA session (it is visible in the CATIA Save Management panel). Perform the sub-steps if it is not:

   a. Open the document in a CATIA session before modifying it.

   b. Save the document with the new full filename provided as a parameter of the *CV5ICAAEntryPoints::entryPointBeforeLocalSave* function if CAA APIs are used or of the CATMain procedure if CATScript is used.

   c. Close the document.

2. Add modifications as needed. It is not necessary to save the document, it will be saved by the Integration.

7.4.2 CATIA Settings Access

Example

You must be in a SWT Display Thread to interact with the Teamcenter user interface such as, get the selected context (selected BOMLine) in Teamcenter, before an entry point can be customized.

Use the single java class, **CatiaOptionsParser**, to access CATIA settings. The javadoc of this class can be found within *{install directory}\custom_client\EntryPoints\javadoc* and is defined within the `com.ebsolutions.catiman.utility` package.

This singleton class provides the capability to access several CATIA setting values managed by the Integration:

Setting	Description
Setting 1 (logical)	CATIA cache activation status
Setting 2 (logical)	CATIA cache timestamp activation status
Setting 3 (logical)	CATIA cache GMT timestamp activation status
Setting 4 (single path)	CATIA local cache path
Setting 5 (list of paths)	CATIA release cache paths

Perform the following steps.

1. Call two methods:

   a. **get_instance()**: Provides the unique instance of the CatiaOptionsParser. The CATIA settings can only be accessed using this method. They cannot be accessed by the CatiaOptionsParser constructor.

   b. **readCatiaOptionsFile()** : Sets several CATIA setting values.

2. Use accessors to retrieve CATIA settings:

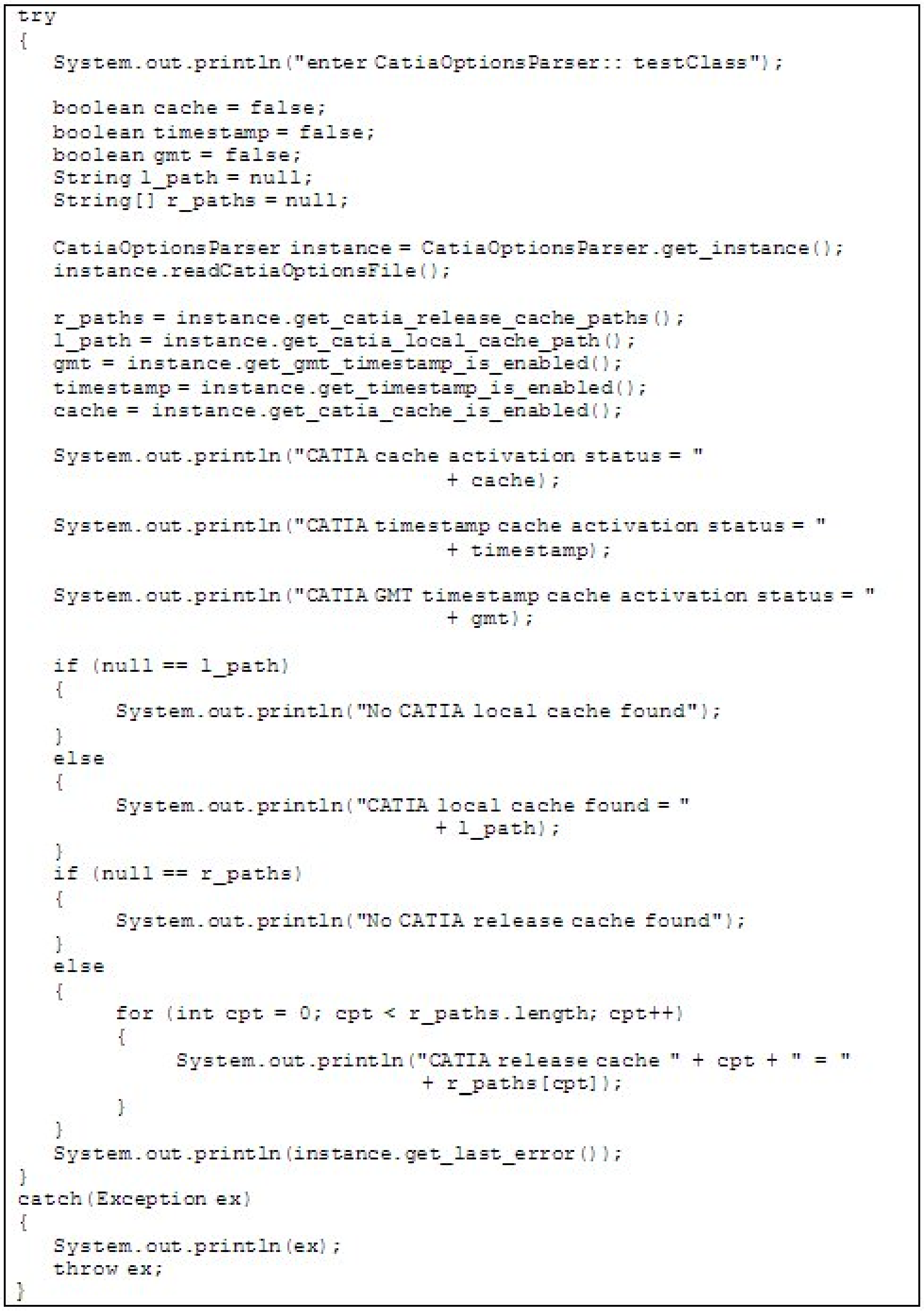
   - **get_catia_cache_is_enabled()**: Accessor to the CATIA cache activation status
   - **get_timestamp_is_enabled()**: Accessor to the CATIA timestamp activation status
   - **get_gmt_timestamp_is_enabled()**: Accessor to t e CATIA GMT timestamp activation status
   - **get_catia_release_cache_paths()**: Accessor to the CATIA local cache directory
   - **get_catia_local_cache_paths()**: Accessor to the CATIA release cache directories.

3. Additional functions:

   - **get_last_error()**: Provides access to the last error occurring during a process. For instance, the error message that occurs when the Integration is unable to read CATIA settings
   - **displaySettings()** : Displays the CATIA settings managed by the Integration.

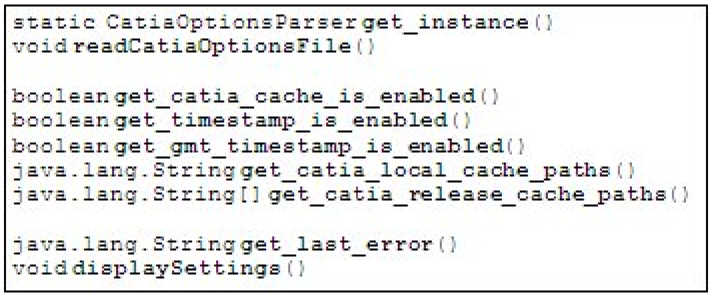
7.4.2.1 Example

Example of a function using the `CatiaOptionsParser`.



```
try
{
   System.out.println("enter CatiaOptionsParser:: testClass");

   boolean cache = false;
   boolean timestamp = false;
   boolean gmt = false;
   String l_path = null;
   String[] r_paths = null;

   CatiaOptionsParser instance = CatiaOptionsParser.get_instance();
   instance.readCatiaOptionsFile();

   r_paths = instance.get_catia_release_cache_paths();
   l_path = instance.get_catia_local_cache_path();
   gmt = instance.get_gmt_timestamp_is_enabled();
   timestamp = instance.get_timestamp_is_enabled();
   cache = instance.get_catia_cache_is_enabled();

   System.out.println("CATIA cache activation status = "
                                + cache);

   System.out.println("CATIA timestamp cache activation status = "
                                + timestamp);

   System.out.println("CATIA GMT timestamp cache activation status = "
                                + gmt);

   if (null == l_path)
   {
        System.out.println("No CATIA local cache found");
   }
   else
   {
        System.out.println("CATIA local cache found = "
                               + l_path);
   }
   if (null == r_paths)
   {
        System.out.println("No CATIA release cache found");
   }
   else
   {
        for (int cpt = 0; cpt < r_paths.length; cpt++)
        {
             System.out.println("CATIA release cache " + cpt + " = "
                                  + r_paths[cpt]);
        }
   }
   System.out.println(instance.get_last_error());
}
catch(Exception ex)
{
   System.out.println(ex);
   throw ex;
}
```

Method Summary



```
static CatiaOptionsParser get_instance()
void readCatiaOptionsFile()

boolean get_catia_cache_is_enabled()
boolean get_timestamp_is_enabled()
boolean get_gmt_timestamp_is_enabled()
java.lang.String get_catia_local_cache_paths()
java.lang.String[] get_catia_release_cache_paths()

java.lang.String get_last_error()
void displaySettings()
```

7.4.3 Load Entry Point

userExitAfterAllExportedFiles Function

userExitAtEndOfLoadProcesses Function

CATIA Side

You must be in a EDT (Event Display Thread) to interact with the Teamcenter user interface such as, get the selected context (selected BOMLine) in Teamcenter, before an entry point can be customized.

The Import/Export process can be extended by other extension points using the `com.ebsolutions.catiman.extensions` extension.

Create an extension point with a class which implements the `ICatiaUserExit` interface. This interface provides entry points in Teamcenter.

The following functions are public members of the `ICatiaUserExit` interface.

- userExitAfterAllExportedFiles function
- userExitAtEndOfLoadProcesses function

This interface can have several implementations. Each implementation is automatically called when defined in *tcic_extensions.xml* file

Implement the **userExitAfterAllExportedFiles** or **userExitAtEndOfLoadProcesses** method to define your own behavior. Refer to the User Interface Implementation: Main Classes section for details on extension point implementations, if necessary.

7.4.3.1 The userExitAfterAllExportedFiles Function

Launching a process on Load can only be performed if the document has been exported. To launch a process when the document has been exported, use the entry point method `userExitAfterAllExportedFiles` in the user's `ICatiaUserExit` class.

This method is called during every load process:

- after the export of all the files from the database.
- after the creation of the .dat files.
- after the update of the files modification date
- after calling the *userExitAfterAllExportedFiles(Vector files_list)* entry point.

The prototype of the function is:

- `userExitAfterAllExportedFiles(Vector files_list)`
  The input is a string vector compound containing the names of the exported files.

-
  ```
  userExitAfterAllExportedFiles(Connection i_connection,
  Vector<string> i_file_name_list, Vector<string> i_item_id_list,
  Vector<string> i_item_rev_list, Vector<string> i_dataset_uid_list)
  ```

All vectors are string vector compounds containing exported file information.

- i_connection (I): current TC session
- i_file_name_list (I): Vector contains a list of exported file names in catuii
- i_item_id_list (I): Vector contains a list of exported Item IDs in catuii
- i_item_rev_list (I): Vector contains a list of exported Item Rev in catuii
- i_dataset_uid_list (I): Vector contains a list of exported Dataset UID in catuii

7.4.3.2 The userExitAtEndOfLoadProcesses Function

Launching a process on Load can be performed even if the document has not been exported. To launch a process when the document has not been exported, use the entry point method `userExitAtEndOfLoadProcesses` in the user's `ICatiaUserExit` class.

This method is called during every load process:

- after the export of all the files from the database.
- after the creation of the .dat files.

- after the update of the files modification date.
- after the transmission of the desc file.

The prototype of the function is `userExitAtEndOfLoadProcesses()`. Parameters are not required.

7.4.3.3 CATIA Side

An entry point exists on the CATIA side after the Load process. The `EntryPoints.CATScript` CATScript must exist in the *$TCICV5_DIR/custom_CATScript* folder to be activated after the Load process.

If the CATScript does not exist in the correct folder or does not contain the `entryPointAfterLoadProcesses()` function, the Load process functions as usual. If the CATScript exists in the correct folder and contains the `entryPointAfterLoadProcesses()` function, the `entryPointAfterLoadProcesses` launches after the Load process in CATIA.

The CATScript **must** contain a `Sub CATMain()` instruction

```
Sub entryPointAfterLoadProcesses()
MsgBox "hello world"
End Sub
```

Processes impacted by this entry point are:

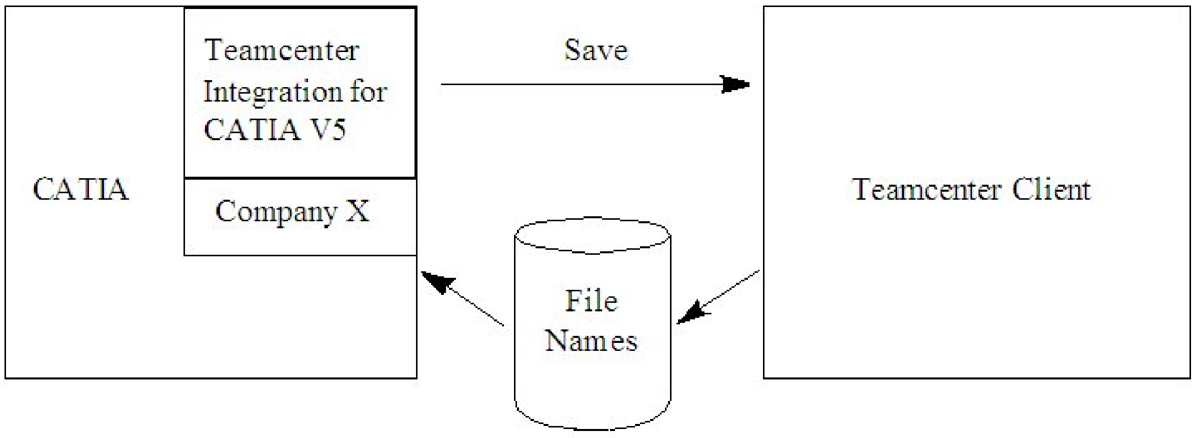
- Browse MML
- Export
- Load.CATScript
- Load_merge.CATScript
- Load_merge_Target
- Load_merge_SelectedLevel
- Refresh
- Read_linked_documents

7.4.4 Save Entry Point

You must be in a EDT (Event Display Thread) to interact with the Teamcenter user interface such as, get the selected context (selected BOMLine) in Teamcenter, before an entry point can be customized.

Use Save Entry/User Exit Points to retrieve a list of saved files in order to apply a post action in CATIA.

These points are managed after the Save process on the Teamcenter Client side and after the Save process on the CATIA side. In addition, an Entry Point/User Exit point can also be managed before the Save process is started in CATIA.



7.4.4.1 CATIA Side

Two entry points exist on the CATIA side; one *before* the Save process and one *after* the Save process. Processes impacted by these entry points are:

- Save

- Save As
- Save Selected
- Import
- Load, Update, Save/Background Round-Robin

7.4.4.1.1 CATIA Side: Before Save Entry Point

The `EntryPointBeforeSaveProcesses.CATScript` CATScript must exist in the *$TCICV5_DIR/custom_CATScript* folder to activate before the Save process.

The CATScript must contain a Sub `CATMain()`.
If this CATScript does not exist in the correct folder or does not contain a `CATMain()` function, the Save processes will function as normal.

If this CATScript exists in the correct folder and contains the `CATMain()` function, the CATMain function executes before the Save process in CATIA.

7.4.4.1.2 CATIA Side: After Save Entry Point

The `EntryPointAfterSaveProcesses.CATScript` CATScript must exist in the *$TCICV5_DIR/custom_CATScript* folder to activate after the Save process.

The CATScript must contain a Sub `CATMain()`.

If this CATScript does not exist in the correct folder or does not contain a `CATMain()` function, the Save processes will function as normal.

If this CATScript exists in the correct folder and contains the `CATMain()` function, the CATMain function executes after the Save process in CATIA, even if errors occur during the Save process.

7.4.4.2 Teamcenter Side/After Save Entry Point

The client entry point, `userExitAfterAllImportedFiles,` is called at the end of each Save process in the user's `TClassCatiaUserExit ICatiaUserExit` class.

- Save
- Save As
- Save Selected
- Import
- Load, Update, Save/Background Round-Robin

The prototype of this function is:

- `userExitAfterAllImportedFiles (Connection i_connection, Vector<string> i_file_name_list, Vector<string> i_item_id_list, Vector<string> i_item_rev_list, Vector<string> i_dataset_uid_list)`
    - i_connection (I): current Teamcenter Session
    - i_file_name_list (I): Vector contains a list of exported file names in catuii
    - i_item_id_list (I): Vector contains a list of exported item ID's in catuii
    - i_item_rev_list (I) : Vector contains a list of exported item revisions in catuii
    - i_dataset_uid_list (I): Vector contains a list of exported dataset UID in catuii

Only files saved to Teamcenter are listed.

This interface can have several implementations. Each implementation is automatically called when defined in *tcic_extensions.xml* file. Refer to the User Interface Implementation: Main Classes section for details on extension point implementations, if necessary.

7.5 Installation Troubleshooting

7.5.1 Error Logs and Preferences

A log file is created during the installation process and for each process launched if problems occur. Check the log files in the final panel of the installer for additional information.

Information contained in the log file includes warning messages, notifications of issues that may have occurred when attempting to obtain information, and critical error messages indicating possible data corruption.

CATScript and CAA traces are gathered in the same journal file. The journal file is enabled by the **CATIA_journal_catia** preference.

7.5.1.1 Activate the Client Log File

Teamcenter must be restarted after modifying preferences.

Use the **CATIA_journal_client** Teamcenter Preference to enable the client journaling option. The default value is **NO**.

Value	Result
YES	Enable **high-level** tracing. Detailed journaling traces are written to a *.syslog* file and an *.xml* file. Tracing includes function calls, errors, comments, return values, etc.
NO (default traces)	Enable **low-level** tracing such as hardware configuration and library versions. Error traces are written to a *.syslog* file.

Individual files named **_client_"userid"_"randomize number".** are created for each user. A *syslog* and *xml* file are created in the *%TEMP%* folder when the server is running on Windows, and in the */tmp* folder on all other operating systems.

7.5.1.2 Activate the Server Log File

Enable Server Journaling

Server Log Directory

Teamcenter must be restarted after modifying preferences.

7.5.1.2.1 Enable Server Journaling

Use the **CATIA_journal_server** Teamcenter Preference to enable the server journaling option, based on three levels. The default value is **NO**.

Value	Result
YES	Enable **high-level** tracing. Detailed journaling traces are written to a *.syslog* file and an *.xml* file. Tracing includes function calls, errors, comments, return values, etc.

Value	Result
NO (default traces)	Enable **low-level** tracing such as hardware configuration and library versions. Error traces are written to a *.syslog* file.
SYSLOG	Export all high-level traces to the original Teamcenter Syslog.

Individual files named *tcic_server_"userid"_"randomize number"_"command".** are created for each user in the *%TEMP%* folder on Windows, and in the */tmp* folder on all other operating systems.

The *tcic_server_*.xml* file is readable, but only available to the Developer. Contact your Developer for access.

7.5.1.2.2 Server Log Directory

Use **CATIA_journal_server_directory** to define the directory path to store the Integration journal server file.

This preference is not taken into account when **CATIA_journal_server** is defined with the value SYSLOG.

There is no default value. When the preference is not defined or not set, traces are stored in *%temp% folder*.

7.5.1.3 Display a Warning Message During the Save Processs

Set **CATIA_display_warning_messages** to **true** to display the warning message, *There are less/more child components for MyComponent in CATIA than in Teamcenter,* during the Save process. Set it to **false** to write the messages in the command log file.

```
CATIA_display_warning_messages = true
```

The default value is **true**.

This preference corresponds to the previous *com.ebsolutions.catia.displayWarningMessages* property defined in the catiaV5properties.properties file, now obsolete.

7.5.1.4 Enable Tracing Events

Use the **CATIA_journal_catia** case-sensitive preference to enable CATScript and CAA tracing events such as macro executions, function calls and errors to be written to the journal file.

The CATIA_journal_catia:

- replaces the **TMCJournal** environment variable used in previous versions of the Integration

- corresponds to the previous *com.ebsolutions.catia.writeTraceFile* property defined in the catiaV5properties.properties file, now obsolete.

```
CATIA_journal_catia = YES
```

The default value is **NO**.

Value	Result
YES	Enable **high-level** tracing. Detailed journaling traces are written to the *journal.syslog* file and *journal.xml* file. Tracing includes function calls, errors, comments, return values, etc.
NO (default traces)	Enable **low-level** tracing such as hardware configuration and library versions. Error traces are written to a *journal.syslog* file.

Setting catia_journal_catia to YES instead of CATIA_journal_catia, causes only the .syslog file to be created; the CATIA_journal_catia preference takes its default value of NO.

Journal Name

The journal name is *tcic_catia_[user]_[random_number]_[command].**.

Journal files are located in the named repository for the environment variables; *%TEMP%* for Windows and *$TMP* for UNIX. By default, journal files are located in \\temp\\ for Windows and */tmp/* for UNIX, when the environment variables are not created. The catia_trace file, now obsolete, was located in *WORK_DIR/tmp*.

The *journal_*.xml* file is readable, but accessible only to the Developer. Contact your Developer for access.

7.6 Teamcenter Logs

Teamcenter log variables should be set as follows to ensure errors, warnings and other important information is captured:

- `TC_JOURNAL = FULL`
- `TC_JOURNAL_LINE_LIMIT = 0`
- `TC_SQL_DEBUG = BJPT`

7.6.1.1 Translation Services

Translation Request Did Not Complete
Troubleshooting catiav5preview

7.6.1.1.1 Translation Request Did Not Complete

dcproxy must be granted access to write to necessary objects in the Access Manager tree. Granting write access is *in addition to* granting dcproxy write/delete access for DispatcherRequests objects.

7.6.1.1.2 Troubleshooting .catiav5preview

Refer to the Verification topic of Translation Services to verify successful deployment of .catiav5preview.

The thumbnail creator is only applicable to Windows platforms and CATIA must be installed on the same machine used to create the thumbnails.

The getmini.exe file exists on the Module component of the Dispatcher (in *Module\Translators\catiav5preview*) after it has been deployed. It is recommended to test this executable file prior to using it through the translation service by executing getmini.exe *full_path_of_a_CATPart_file_to_translate name_of_the_bmp_ouput_file*. If successful, a .bmp file is created.

```
Execute getmini C:\test\TestPart.CATPart TestPartOutput.bmp
```

7.6.1.2 UNIX/Windows Over the Web (OTW) Deployment Error Messages

Error messages may display for the following possible reasons:

1. The CATIA environment was not found.

    - Creation failed: catenv does not exist or no catenv was selected.

2. CAA binaries not found and the CAA mode was chosen.

    - A path is not valid because a special character was used in the Windows/UNIX path, or a [space] was used in a UNIX path.

8.0 The Preferences .xls File and Configuring the Interface

8.1 Preferences

A complete list of preferences is available in .xls format within the online documentation. Use this file as a quick reference to search and sort for specific preferences.

Please note the .xls file contains basic information such as name and default values, and should only be used as a quick reference. Refer to the documentation for detailed definitions and specific behavior.

Teamcenter must be restarted after modifying preferences.

8.2 Configuring the Interface

There are two different ways to customize and modify the behavior of the interface:

- Teamcenter Preferences
- tcic_var_env.xml file

8.2.1 Teamcenter Preferences

Pick **Edit >> Options...**, then pick **Index** on the Options dialog window to customize all Teamcenter Preferences.

8.2.2 Customize the tcic_var_env.xml File

The default variables in *tcic_var_env.xml* (in the *%TCICV5_DIR%/env/* directory) are:

⚠ ***If these variables are modified, restart Teamcenter to apply the new values.***

Name	Default value	Purpose
TcIC_assyconfig_menu	false	Display Save Assembly Configuration command in the CATIA menu in Teamcenter, in the contextual menu of a selected BOM Line in the Teamcenter Loader and in the Load Manager (true). Display the Activate Terminal Node command in the CATIA toolbar. (true)
TcIC_awhosted_menu	false	Display (true) or hide (false) the TcIC menu when Embedded Active Workspace installation is defined.
TcIC_catalog_dir	*%WORK_DIR%\catalog (WIN)*	Directory where Catalog files and related documents are stored

Name	Default value	Purpose
TcIC_config_dir	*%USERPROFILE%* (WIN)	TcIC filepath for configuration files.
TcIC_dr1_license	false	Support TcIC functionalities (true) when a DR1 license is used.
TcIC_export_dir	*%WORK_DIR%\Export* (WIN)	**Used only by CATIA** Export filepath
TcIC_export_spreadsheet_path	*%WORK_DIR%\Export\ExpSpreadSheet.txt* (WIN)	SpreadSheet file path for EXPORT process
TcIC_files_dir	*%WORK_DIR%\catuii* (WIN)	Directory where product, part, drawing and model files are stored
TcIC_import_data_path	*%WORK_DIR%\Import\Data.txt* (WIN)	**Used only by CATIA** Data filepath
TcIC_import_dir	*%WORK_DIR%\Import* (WIN)	**Used only by CATIA** Import filepath

Name	Default value	Purpose
TcIC_import_spreadsheet_path	*%WORK_DIR%\Import\ImpSpreadSheet.txt* (WIN)	Spreadsheet file path for IMPORT process
TcIC_plmvis_license	Base	PLM Vis license used when displaying the Teamcenter Loader's JT Viewer. Possible values are Base, Standard, Pro and Mockup **NOTE**: Teamcenter Integration for CATIA officially only supports Base license.
TcIC_rac_socket_port	11401	TCP port number used for the interface communication between RAC and RACLess.
TcIC_socket_port	11400	TCP port number for the interface communication and the progress indicator.
TcIC_tmp_dir	*%WORK_DIR%\tmp* (WIN)	Directory where temporary communication files are stored

Name	Default value	Purpose
TcIC_web_server_port	3000	TCP port number used for the Active Workspace web server

Refer to Configuration Directory, Files and Folder for more information on *TcIC_config_dir*.

8.3 The Configuration Directory, Folder and Files

File Names and Descriptions

Location and Naming Convention

8.3.1.1 File Names and Descriptions

Configuration files used to manage processes and components are stored within TcIC_config_dir. Refer to the table below for their names and descriptions.

Name	Type	Description	If the configuration files cannot be created or read...
bbml_user.properties	text	Stores information related to the Browse Multi-model Links browser panel.	---
datasetbrowser_<checksum>.xml	.xml	Stores information related to dataset browser customizations. <checksum> is the checksum of:*<TCICV5_DIR>\ env\ datasetbrowser.xml*	a warning displays

Name	Type	Description	If the configuration files cannot be created or read...
exporttodirectory_<checksum>.xml	.xml	Stores information related to the Export to directory customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ exporttodirectory.xml*	a warning displays
itemrevbrowser_<checksum>.xml	.xml	Stores information related to item revision browser customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ itemrevbrowser.xml*	a warning displays
progressbar	text	Contains the progress panel position. The settings from this file are read at process start and written to disk when the progress bar closes. If the configuration file cannot be read or created, the default position (top-left corner) will be used without warning or notifications.	there is no warning or notification and the progress bar displays in the default position (upper left corner).
replacebyrevisionpanel_<checksum>.xml	.xml	Stores information related to replace by revision panel customizations.<checksum> is the checksum of: <TCICV5_DIR>\env\ replacebyrevisionpanel.xml	a warning displays
savemanager<checksum>.xml	.xml	Stores information related to Save Manager panel customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ savemanager.xml*	a warning displays

Name	Type	Description	If the configuration files cannot be created or read...
sprdshteditor_<checksum>.xml	.xml	Stores information related to Spreadsheet Editor panel customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ sprdshteditor.xml*	a warning displays
statedetails_<checksum>.xml	.xml	Stores information related to State Details panel customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ statedetails.xml*	a warning displays
seedpanel_<checksum>.xml	.xml	Stores information related to Seed panel customizations. <checksum> is the checksum of: *<TCICV5_DIR>\ env\ seedpanel.xml*	a warning displays

8.3.1.2 Location and Naming Convention

The location of the configuration folder is set using the value of TcIC_config_dir , defined in the *tcic_var_env.xml* file.

The configuration folder is created in *<TcIC_config_dir >* + *file separator* + *.TcIC*, based on the following rules:

- The name of the configuration folder is built using the value of the TCICV5_DIR environment variable
- File separators (**/** and ****) are replaced by underscores (_).'
- The colon (**:**) is replaced by a dash (**-**).

For example:

On Windows: If *TCICV5_DIR* = *C:\TcIC*, then the directory name is *C-_TcIC* and the complete path is *<TcIC_config_dir >\.TcIC \C-_TcIC.*

On UNIX: If *TCICV5_DIR = /tmp/TcIC*, then the directory name is *_tmp_TcIC* and the complete path is *<TcIC_config_dir >/.TcIC /_tmp_TcIC.*

9.0 Directories and Filepaths, Environment Variables and Parameter Definitions Table

9.1 Directories and Filepaths

The default variables in *tcic_var_env.xml* (in the *%TCICV5_DIR%/env/* directory) are:

If these variables are modified, restart Teamcenter to apply the new values.

Name	Default value	Purpose
TcIC_assyconfig_menu	false	Display Save Assembly Configuration command in the CATIA menu in Teamcenter, in the contextual menu of a selected BOM Line in the Teamcenter Loader and in the Load Manager (true). Display the Activate Terminal Node command in the CATIA toolbar. (true)
TcIC_awhosted_menu	false	Display (true) or hide (false) the TcIC menu when Embedded Active Workspace installation is defined.
TcIC_catalog_dir	*%WORK_DIR%\catalog (WIN)*	Directory where Catalog files and related documents are stored

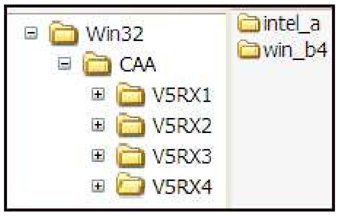
Name	Default value	Purpose
TcIC_config_dir	*%USERPROFILE%* (WIN)	TcIC filepath for configuration files.
TcIC_dr1_license	false	Support TcIC functionalities (true) when a DR1 license is used.
TcIC_export_dir	*%WORK_DIR%\Export* (WIN)	**Used only by CATIA** Export filepath
TcIC_export_spreadsheet_path	*%WORK_DIR%\Export\ExpSpreadSheet.txt* (WIN)	SpreadSheet file path for EXPORT process
TcIC_files_dir	*%WORK_DIR%\catuii* (WIN)	Directory where product, part, drawing and model files are stored
TcIC_import_data_path	*%WORK_DIR%\Import\Data.txt* (WIN)	**Used only by CATIA** Data filepath
TcIC_import_dir	*%WORK_DIR%\Import* (WIN)	**Used only by CATIA** Import filepath

Name	Default value	Purpose
TcIC_import_spreadsheet_path	*%WORK_DIR%\Import\ImpSpreadSheet.txt* (WIN)	Spreadsheet file path for IMPORT process
TcIC_plmvis_license	Base	PLM Vis license used when displaying the Teamcenter Loader's JT Viewer. Possible values are Base, Standard, Pro and Mockup **NOTE**: Teamcenter Integration for CATIA officially only supports Base license.
TcIC_rac_socket_port	11401	TCP port number used for the interface communication between RAC and RACLess.
TcIC_socket_port	11400	TCP port number for the interface communication and the progress indicator.
TcIC_tmp_dir	*%WORK_DIR%\tmp* (WIN)	Directory where temporary communication files are stored

Name	Default value	Purpose
TcIC_web_server_port	3000	TCP port number used for the Active Workspace web server

9.2 Parameter Definitions Table

The paths are case-sensitive.

Parameter	Type	Default Value	Description
CAA_Binaries_Unix_Location	STR	/CAA_Binaries	UNIX CAA location. Folder structure: CAA_Binaries_Win_Location/<OS>/Caa/V5Rxx (<*OS*> is SunOS, AIX or HP-UX, *xx* is the version of CATIA V5)
CAA_Binaries_Win_Location	STR	C:\CAA_Binaries	Windows CAA location. Folder structure: CAA_Binaries_Win_Location/ Win32/Caa/V5RXX (*XX* is the CATIA V5 release number). The same source directory, \Win32, is used for **both** 32 and 64-bit deployments. Differences between 32 and 64-bit are made at the \V5RXX folder level. The expected structure is shown below; **\intel_a** contains the 32-bit compiled CAA and **\win_b64** contains the 64-bit compiled CAA.  Win32 CAA V5RX1 V5RX2 V5RX3 V5RX4 intel_a win_b4

Parameter	Type	Default Value	Description
CATEnv_Name	STR	Teamcenter_Integration_for_CATIAV5	CATEnv name to be used by Teamcenter Integration for CATIA V5. Indicate *No CATEnv Creation* if an existing CATEnv will be used.
CATIA_Env_Creation	STR	Create CATEnv	*Create CATEnv* generates the CATIA environment named *${CATEnv_Name}*
CatiaV5_Env_Unix_location	STR	/CATEnv	CATEnv location on UNIX client machines. This value must be the same across all UNIX client machines.
CatiaV5_Env_Win_location	STR	C:\Documents and Settings\All Users\Application Data\DassaultSystemes\CATEnv	CATEnv location on Windows client machines. This value must be the same across all Windows client machines.
catiaV5_integration_TmpDir_Unix_location	STR	/tmp/catiav5_integration_tmp_${LOGNAME}	Location of the Teamcenter Integration for CATIA V5 temporary files on UNIX client machines *WORK_DIR*. This value must be the same across all UNIX client machines.
catiaV5_integration_TmpDir_Win_location	STR	%TEMP%\catiav5_integration_tmp	Location of the Teamcenter Integration for CATIA V5 temporary files on Windows client machines *WORK_DIR*. This value must be the same across all Windows client machines.

Parameter	Type	Default Value	Description
CatiaV5_Unix_location	STR	/opt/DassaultSystemes/B18	CATIA V5 location on UNIX client machines. This value must be the same across all UNIX client machines.
CatiaV5_Unix_Version	INT	18	CATIA V5 version on UNIX client machines. It must be the same version across all UNIX client machines.
CatiaV5_Win_location	STR	C:\Program Files\Dassault Systemes\B18	CATIA V5 location on Windows client machines. This value must be the same across all Windows client machines.
CatiaV5_Win_Platform	STR	32 bits	Defines if CATIA V5 is 32 or 64 bits. This value must be the same across all Windows client machines.
CatiaV5_Win_Version	INT	18	CATIA V5 version on Windows client machines. It must be the same version across all Windows client machines.
Tc_Version	STR	---	Teamcenter version number
TcIC_Documentation_Action	STR	Install Documentation	Used with Over the Web (OTW) installations. Defines whether or not to install the user documentation.

Parameter	Type	Default Value	Description
TcIC_Documentation_Location_UNIX	STR	<empty>	Location of the user documentation on UNIX.
TcIC_Documentation_Location_Windows	STR	<empty>	Location of the user documentation on Windows.
TcIC_Reference_UNIX_Location	STR	$WORK_DIR/reference	Location of the Teamcenter Integration for CATIA V5 reference files on UNIX client machines.
TcIC_Reference_Win_Location	STR	%WORK_DIR%\reference	Location of the Teamcenter Integration for CATIA V5 reference files on Windows client machines.
UnixLocation	STR	/Siemens/Teamcenter/OTW8	Name of the Teamcenter Client directory (UNIX)
WindowsLocation	STR	C:\Program Files\Siemens\Teamcenter\OTW8	Name of the Teamcenter Client directory (Windows)

9.3 System and User Environment Variables

Change system and user environment variables to modify the behavior of the interface. By default the system environment variables are:

Name	Default value on a PC	Default value on UNIX	Purpose
PURGE_MSG	both	both	Controls the Purge dialog box
TC_PORTAL_ROOT	STRING	C:\Tc_Version\portal	Teamcenter Rich client directory
TC_ROOT	STRING	C:\Tc_Version	Teamcenter server directory
TCICV5_DIR	STRING	C:\TCIC_V5	Teamcenter Integration for CATIA V5 installation location
WORK_DIR	STRING	C:\TCIC_V5_tmp	Teamcenter Integration for CATIA V5 temporary files location

10.0 Uninstall Teamcenter Integration for CATIA V5

Removing the JVM, installed by default, causes the client uninstallation process to fail.

10.1 Uninstall Teamcenter Integration for CATIA V5 (Client)

Removing the JVM installed by default causes the client uninstallation process to fail.

The uninstall process does not remove Integration plug-in files from Teamcenter's RAC. Before removing Integration plug-in files, manually sort the files within the portal\plugins directory by using the cat2 prefix. It is not necessary to remove the Cat2ebscomservice.jar file when Teamcenter Integration for CATIA V4 is also used because it is common to both Teamcenter Integration for CATIA V4 and Teamcenter Integration for CATIA V5.

Refer to the following steps to uninstall Teamcenter Integration for CATIA V5 when it has been installed on a Client.

1. Launch the Client uninstaller:

    a. **Windows**: Double-click on *Teamcenter Integration for CATIA V5 <version number> Client* within the *Add or Remove Programs* panel.

    b. **All other installations**: Use the *<Integration Client location>/Uninstall/Uninstall-TCIC-Client* command.

2. Read the uninstallation message in the Uninstall window to ensure you are ready to proceed, then pick **Uninstall** to continue. Please wait a moment for the uninstallation process to begin.

3. Read the uninstallation successful message, then pick **Done**. Uninstallation is now complete.

10.2 Uninstall Teamcenter Integration for CATIA V5 (Server)

Refer to the following steps to uninstall Teamcenter Integration for CATIA V5 when it has been installed on a Server.

1. Launch the Server uninstaller:

   a. **Windows**: Double-click Teamcenter Integration for CATIA V5 <*version number*> Server within the *Add or Remove Programs* panel.

   b. **All other installations**: Use the <*Teamcenter Server location*>/Uninstall_TCIC_Server/Uninstall_TCIC_Server command.

2. Read the uninstallation message in the Uninstall window to ensure you are ready to proceed, then pick **Uninstall** to continue. Please wait a moment for the uninstallation process to begin.

3. Read the uninstallation successful message, then pick **Done**. Uninstallation is now complete.

10.3 Uninstall Teamcenter Integration for CATIA V5 (Silent Mode)

Launch the command line, `%TCICV5_DIR%\Uninstall\Uninstall-TCIC-Client.exe –i silent` to initiate the silent uninstaller.