Siemens PLM Software

Teamcenter Gateway 18.2

RAC GUI – Configuration Examples

About this document

This document explains how to configure the T4x 18.2 GUI using Teamcenter preferences. Examples are given as preference files so they can be imported into Teamcenter. You may have to modify the examples to suit your needs.

The software is continually being enhanced. We reserve the right to add or modify functionality in a way that is not consistent with the description in this manual as part of the routine maintenance and enhancement process. We will do our best to keep this manual up to date to reflect those changes.

Issue: November 2018

Legal notice:

Trademark notice:

Documentation Conventions:

The manual uses the following conventions:

Note

- A note gives important hints and technical background information and helps you taking decisions later.

Recommendation

- If several choices exist for an installation or operation step, we recommend a solution that works best in most configurations. However you may choose another alternative depending on your environment.

Caution

- Cautions are important notes with crucial impact on the installation or operation of the software

Prerequisites

- Prerequisites describe conditions that must be fulfilled before installation or operation can continue

Validation

- After some installation or operation step you should validate the result as described here before proceeding

Table of Contents

1 How to use this Example Collection

This document comes with examples given as the content of Teamcenter preference files. You can modify them to suit your needs and then import them into the Teamcenter Rich Client (RAC).

The examples assume that you have installed Teamcenter 10.1.0 or higher. You can use them for older Teamcenter versions if you remove two attributes from the XML:

```
disabled="false" protectionScope="Site" envEnabled="false"
```

The installation of T4x includes importing an initial set of preferences into Teamcenter. The preference files presented here provide examples that go beyond that.

The names of GUI preferences have prefixes to control to which T4x flavor they apply. At the time of this writing the flavor-specific prefixes are “T4S.”, “T4O.” or “T4EA.” (without the quotes). The “T4X.”, prefix applies to all flavors of T4x at the same time.

Caution

- There are more products in the T4x family, e.g. T4CPG. However as they do not have a GUI component they are not covered here.
- Some of the examples cannot be applied to T4EA as it does not have a transaction window and does not support custom actions.

Note

- In the examples the “T4X.”, prefix is used for the preferences that could be applied to more than one flavor with the same semantics. Preferences which are specific for a particular flavor are prefixed with the flavor-specific string.
- If we ask you to modify a preference and you find that this preference does not exist yet just create it.
- *Italic font* is used to mark text strings that act as placeholders or for references to other parts of the document.
- When editing preference files and using non ASCII characters be careful to adjust your text editor to save the file in the encoding specified in the first line of the file, e.g. UTF-8.

2 Main Principles for GUI Preferences

We recommend reading this chapter in order to understand how preferences are named and what sub preferences are.

2.1 Think of objects and attributes

Preferences describe objects, for example menu items. Objects can contain or reference other objects building up a hierarchy like the files, directories and links in your computer's file system.

You sometimes will want to create new objects using preferences, for example new menu items inside the T4x Gateway menu. You then have to invent IDs for them so they can be referenced from other preferences. The latter you need to specify the object's details.

There are also objects you can point to that you do not need to create, for example predefined menu items.

2.2 Preference names

The names of GUI preferences are like the paths in a file system. However dots instead of slashes or backslashes are used to separate their elements.

The leftmost elements are T4S, T4O, T4EA or T4X. They denote the T4x GUI itself. Use T4S, T4O or T4EA to limit the scope of the preference to that flavor. T4X stands for any of the first three.

The rightmost elements of the paths are the names of attributes.

The values of attributes can be IDs. Depending on the semantics of the attribute they cause the creation of new objects or point to objects created through other preferences.

In order to reduce the number of preferences you have to create there are attributes whose values are implicitly set to their object's ID if not overridden by another preference. For example if you do not provide a displayable name for a menu item its ID is used.

The elements in the middle represent chains of references from one object to another. Sometimes references need to denote objects you created. Then the references are followed by the object's IDs.

A preference S is called **sub preference** of a preference P if S's name is P's name plus a dot and a string with no dot.

You will see how that works in the examples.

2.3 Template preferences

Some attribute values will undergo template expansion before they are actually used. Placeholders of the form `${Name}` will be substituted with the value of the variable ***Name*** if it identifies a valid variable. Otherwise the placeholder remains in the value of the attribute.

Variable names are actually dot-separated paths, with the first elements naming the *sources* of the variables. For example `${env.TPR}` would be expanded to the value of the TPR environment variable.

The following sources are available:

env

Environment variables, for example `${env.TPR}` would expand to `C:\UGS\tc101\portal`. (This is not related to env-enabled preferences where the name of the preference is taken as the name of an environment variable)

prop

Java system properties, for example `${prop.osgi.instance.area}` would expand to `file:/C:/Users/infodba/Teamcenter/RAC/20120329163309/`.

textServer

Text server entries, for example `${textServer.web_initproc_title}` would expand to "New Process" in an English environment. You can add your own entries. See below in the *2.5* section.

pref.default

Values of preferences where the values of array preferences are separated by newline characters

appData

Values stored in the data model of the GUI, for example `${appData.productTag}` would expand to the name of the flavor, or `${appData.eaShortName}` to SAP, EBS, etc.

Note that just after starting the Teamcenter Rich Client the GUI logs the available environment variables and Java system properties if you configured it accordingly.

2.4 Override texts with preferences

In order to allow users of different languages to use the GUI in their own languages all texts are also stored in language-specific Java properties files. They can be overridden through Teamcenter preferences as well. If you know the name of the property just prefix it with `T4S.`, `T4O.`, `T4EA.` or `T4X.` to get the name of the preference. In order to find the property you will have to open one of the jar files whose names start with `de.tesis.plmware` and end with `app.jar`. Inside them check out the properties folder and look into the files whose names start with `texts` and end with a language key and `.properties`.

2.5 Internationalize overridden texts

The Teamcenter text server maintains sets of key-value pairs to be able to label UI elements in various languages. The keys can be used in placeholders of the form `${textServer.KEY}`. The placeholders will be replaced with the values bound to the keys. A single key can denote different values each for a particular language. There are XML files that define all the key-value pairs. They are normally located under `$TC_ROOT/lang/textserver` in directories named after the locale to which the language belongs.

Customers can add their own key-value pairs in files named `user_property_names.xml`. For example the French key-value pairs would be added to the `$TC_ROOT/lang/textserver/fr_FR/user_property_names.xml` file.

Look into the existing language files and create your own key-value pairs using copy&paste.

Caution

- Be careful to create keys that are not in use already. We recommend prefixing them with a string identifying your customization.

Changes to the language files need to be propagated to the Teamcenter clients. Open a command shell with Teamcenter environment and run the following command:

```
generate_client_meta_cache -u=infodba -p=password -g=dba update textservers
```

You need to replace *password* with the password for the infodba user. For more information search the Teamcenter documentation for `generate_client_meta_cache` or navigate to |

Home > *Administering Teamcenter* > *Utilities Reference* > *Maintenance utilities* > *System maintenance*.

2.6 Compatibility layer

The current T4x GUI replaces an older implementation that used to use a different scheme for the preference names. The older naming scheme is still used by the T4x server.

In order to stay compatible with existing configurations and with the T4x server the GUI maps preference names to alternate names and in some cases even translates their values so they can be evaluated by the current GUI.

According to old scheme names start with `T4X_`, `T4S_`, `T4O_` or `T4EA_`. Note that these prefixes are fix, so if you know about a preference that works for T4S you cannot simply make it available to all flavors by simply replacing `T4S_` with `T4X_`.

Where the new names as well as the old names are allowed only the new names will be given. If for compatibility reasons only the old names are allowed, e.g. because the T4x server needs them, only the old names are given.

3 Preferences covering the Gateway menu

Note that custom menu items are covered in chapter 5 Custom menu items on page 16

3.1 Hide the Gateway menu

In order to prevent the Gateway menu from appearing in the main menu of the Teamcenter Rich Client just set the `T4X.UI.GatewayMenu.Hide` preference to true.

This example demonstrates how to prevent the Gateway menu from appearing in the main menu of the Teamcenter Rich Client:

hide_gateway_menu.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.GatewayMenu.Hide" type="Logical"
            array="false" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference prevents the Gateway menu
                from appearing in the main menu of the Teamcenter Rich Client
            </preference_description>
            <context name="Teamcenter">
                <value>true</value>
            </context>
        </preference>
    </category>
</preferences>
```

3.2 Control in which Application the Gateway menu will be visible

Technically speaking Teamcenter applications are Eclipse perspectives.

The GUI changes the visibility of the Gateway menu according to the current perspective.

The list of perspectives which are to contain the Gateway menu is specified through the `T4X.UI.GatewayMenu.VisibleIn` array preference. If the preference is not defined the following default will be used:

- com.teamcenter.rac.ui.perspectives.navigatorPerspective
- com.teamcenter.rac.pse.PSEPerspective
- com.teamcenter.rac.cme.pmp.PMPPerspective
- com.teamcenter.rac.cme.collaborationcontext.CollaborationContextPerspective
- com.teamcenter.rac.cme.mpp.MSEPerspective
- com.teamcenter.rac.cm.perspectives.changeManager

These list items are perspective IDs. They are independent of the language in which Teamcenter is run. In order to find out the ID of a particular perspective you can enable GUI logging, change to the perspective whose ID you are interested in and watch the log output.

You will see a message like

```
DEBUG[main ]: Perspective changed:
DEBUG[main ]: id: com.teamcenter.rac.pse.PSEPerspective
DEBUG[main ]: label: Structure Manager
```

You can also address perspectives using their labels but keep in mind that they depend on the language in which Teamcenter was launched.

You can use asterisks to specify patterns to be matched against the name or IDs of perspectives, e.g. com.teamcenter.rac.cme.*

If you specify `T4X.UI.GatewayMenu.VisibleIn` to be empty the default will be used.

In order to not having to repeat the default perspectives there are two more preferences you may use:

T4X.UI.GatewayMenu.AlsoVisibleIn

The items in this list are also used to check whether the current perspective shows the Gateway menu or not.

T4X.UI.GatewayMenu.HiddenIn

The items in this list are used to specify perspectives in which to hide the Gateway menu. It subtracts entries from the `T4X.UI.GatewayMenu.VisibleIn` (or its default if it does not exist) and `T4X.UI.GatewayMenu.AlsoVisibleIn` preferences . Use it to reduce the default set of perspectives.

3.2.1 Example: Show Gateway menu in “Classification” but not in “Change Manager”

gateway_menu_for_application.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.GatewayMenu.HiddenIn" type="Logical"
            array="true" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the applications which
                will not contain the Gateway menu
            </preference_description>
            <context name="Teamcenter">
                <value>*.changeManager</value>
            </context>
        </preference>
        <preference name="T4X.UI.GatewayMenu.AlsoVisibleIn" type="Logical"
            array="true" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the applications which
                will contain the Gateway menu
            </preference_description>
            <context name="Teamcenter">
                <value>*.ClassificationPerspective</value>
            </context>
        </preference>
    </category>
</preferences>
```

3.3 Change the set of predefined menu items in the Gateway menu

The Gateway menu consists of three sections:

1. EA object types (more in chapter *4 Available EA object types* on page 15)
2. Custom menu items (more in chapter *5 Custom menu items* on page 16)
3. Predefined menu items

This is about the last section. Only the menu item to show the connections to the EA systems is available out of the box. You can change this by modifying the `T4X.PredefinedMenuItems` preference. Just enter the IDs of the menu items you want to appear in the Gateway menu. The IDs are as follows:

EaConnections (SAP_CONNECT, EBS_CONNECT, etc., whatever ends with CONNECT)

Menu item to open the window showing the connections to the EA systems

ShowLog (SHOW_LOG)

Menu item to open the window showing the logging information for the selected Teamcenter object

ShowCustomData

Menu item to open the data view; this serves as a short cut for Window → Show View → Other... → Teamcenter Gateway → SAP Data

The values in parentheses are also valid for compatibility reasons.

3.3.1 Example: Show all predefined menu items

This example demonstrates how to enable all predefined menu items in the Gateway menu.

show_all_predefined.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.PredefinedMenuItems" type="String"
            array="true" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies which predefined
                menu items to enable in the Gateway menu.
            </preference_description>
            <context name="Teamcenter">
                <value>ShowCustomData</value>
                <value>SHOW_LOG</value>
                <value>CONNECT</value>
            </context>
        </preference>
    </category>
</preferences>
```

4 Available EA object types

Each flavor of T4x comes with a predefined set of EA object types (also called target types), e.g. T4S comes with Material Master (MM), Bill Of Material (BOM), Document Info Record (DIR), etc. Admins could completely redefine them modifying the `T4X.EaObjectTypes` preference but normally they want to add new EA object types. In order to avoid them having to re-specify the predefined EA object types, new types can be added via the `T4X.NewEaObjectTypes` preference.

4.1 Example

For the examples assume the admin wanted to add the Schedule type.

new_ea_object_types.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.NewEaObjectTypes" type="String"
            array="true" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference adds new EA object
                types to the T4x GUI.
            </preference_description>
            <context name="Teamcenter">
                <value>Schedule</value>
            </context>
        </preference>
    </category>
</preferences>
```

4.2 Compatibility

In order to ease sharing parts of the configuration between the various flavors of T4x we tried to find names for the EA object types which abstract from the EA system. For example the SAP-specific name "Document Info Record" became "Document". We call them generic names and use them for the IDs of the EA object types. The names which are specific for the EA systems remain existent in the displayName, wireName and shortWireName attributes. For example for the "Document" type the SAP-specific names are "Document Info Record", "DocumentInfoRecord" and "DIR" resp.

However in cases where the communities around the various EA systems have strongly gotten used to particular names we went back to more EA-specific names. For example the terms "Material", "Item" and "Article" describe similar concepts in T4S, T4O and T4EA resp.

5 Custom menu items

In order to demonstrate how to add your own items to the Gateway menu we are starting with the simplest example and evolve this towards more complex ones.

5.1 Add menu item at its default position

5.1.1 Add the menu Item

In order to add a new menu item to the Gateway menu simply invent an ID for it and add it to the `T4X.CustomMenuItems` preference. The ID will be added to the menu just below the automatically generated items for the EA object types. It is still disabled i.e. clicking on it has no effect and it appears grayed out. This will change as soon as we connect it with some sort of operation. See *5.4 Assign a custom action to a custom menu item* on page 22.

5.1.2 Give it a name other than its ID

In order to give a menu item another name than its ID create the DisplayName sub preference and set it to the desired name. This is a template preference. Consult the *Template preferences* section on page 9 to learn more about them.

5.1.3 Give it an icon

In order to give a menu item an icon other than its ID, create the Icon sub preference and set it to the desired path. This is a template preference. Consult the *Template preferences* section on page 9 to learn more about them.

5.1.4 Add a separator to the Gateway menu

A separator is a horizontal line you can add to the menu to visually separate your custom menu items from the automatically generated ones or from each another. A menu item becomes a separator if it is given the special ID `:separator` . In contrast to regular IDs this one can appear more than once among the custom menu items.

5.1.5 Example: create a menu item with a display name and an icon

This example demonstrates the matters given in the above sections.

For the Icon sub preference we copied the `newprocess.png` file out of Teamcenter to the directory where the Teamcenter Rich Client is installed. The following icon will then appear on the left side of the menu item's name:

add_menu_item.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String" array="true"
            disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level custom
                menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>MyMenuItem</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MyMenuItem.DisplayName"
            type="String" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies a display name for
                a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>My Menu Item</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MyMenuItem.Icon" type="String"
            array="false" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the path of an
                icon for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>${env.TPR}/newprocess.png</value>
            </context>
        </preference>
    </category>
</preferences>
```

You may try this with any png file.

Import the preference file with

Merge preference values in the database with values in the XML files

checked to repeatedly add items to the Gateway menu.

5.2 Add menu item at a particular position inside the Gateway menu

In order to place menu items at particular positions inside the menu, e.g. at the top of the menu, add the Position sub preference to the menu item. The IDs of separators you want to place at particular positions need to be changed to allow the separator to be uniquely identified. Just prefix the `:separator` IDs with a unique name, e.g. Sep1.

5.2.1 Example: position a menu item

This example demonstrates how to place the separator and the menu item introduced in the previous example at the top of the menu item.

positioned_menu_items.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String" array="true"
            disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level custom
                menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>Sep1:separator</value>
                <value>MyMenuItem</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.Sep1:separator.Position"
            type="Integer" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies a position for a
                custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>2</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MyMenuItem.DisplayName"
            type="String" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies a display name for
                a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>My Menu Item</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MyMenuItem.Icon" type="String"
            array="false" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the path of an
                icon for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>${env.TPR}/newprocess.png</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MyMenuItem.Position"
            type="Integer" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies a position for a
                custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>2</value>
            </context>
        </preference>
    </category>
```

```
</preferences>
```

5.3 Add a sub menu with a menu item to the Gateway menu

A custom menu item becomes a sub menu if you add the IDs of the subordinate menu items to its SubItems sub preference.

5.3.1 Example: Create a menu item inside a sub menu

This example demonstrates how to place the menu item introduced in the first example inside a subordinate menu.

sub_menu.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String"
            array="true" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level
                custom menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>MySubMenu</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MySubMenu.DisplayName" type="String"
            array="false" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies a display
                name for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>My Sub Menu</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.MySubMenu.SubItems" type="String"
            array="true" disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the items of a
                subordinate menu.
            </preference_description>
            <context name="Teamcenter">
                <value>MyMenuItem</value>
            </context>
        </preference>
        <preference
            name="T4X.CustomMenuItems.MySubMenu.SubItems.MyMenuItem.DisplayName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies a display
                name for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>My Menu Item</value>
            </context>
        </preference>
        <preference
            name="T4X.CustomMenuItems.MySubMenu.SubItems.MyMenuItem.Icon"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the path
                of an icon for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>${env.TPR}/newprocess.png</value>
            </context>
        </preference>
```

```
    </category>
</preferences>
```

5.4 Assign a custom action to a custom menu item

Up to now we covered the representation of the menu items. Now we want to connect them with some sort of operation. We start with so-called custom actions. They can be used to create new workflow processes or invoke Tcl procedures defined in `.sd` files under `var/mmap`.

See chapter *6 Preferences to specify (custom) actions* on page 27.

5.5 Assign a command to a custom menu item

The Eclipse framework which the Teamcenter rich client is based on allows the definition of operations as so-called commands. Simply speaking commands have a command id used to reference them and are associated with a Java class or object to perform the operation. For example Teamcenter allows the invocation of commands from within the summary view. The GUI creates such commands for the common actions such as Create, Create Direct, etc. and some other operations not related to actions.

In order to connect a custom menu item to a command create the Command sub preference plus the CommandId sub preference and set the latter to the ID of the command to be invoked through the custom menu item.

Command IDs are typically dot separated paths starting with the name of the plug-in in which they are implemented. They are specified in the plugin.xml files inside the Jar files below the `portal/plugins` directory of the Teamcenter installation directory.

5.5.1 Example: create a new workflow process through a command

This example demonstrates how to invoke a parameter-less command through a custom menu item. The command opens the dialog to create a new workflow process. Note that it does not allow the specification of the details of the process such as the name of the workflow template.

If you simply want to create a process by-passing the dialog you need to do so using custom actions with WORKFLOW mode.

See chapter *6 Preferences to specify (custom) actions* on page 27.

new_process_command_menu_item.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String"
            array="true" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference lists the top level
                custom menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>NewProcess</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewProcess.DisplayName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies a display
                name for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>Create a new Workflow Process</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewProcess.Command.CommandId"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the
                command id of the command to be invoked through this
                custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>com.teamcenter.rac.newProcess</value>
            </context>
        </preference>
    </category>
</preferences>
```

Note that most commands bring their own icon so you do not need to specify one for the menu item.

5.6 Pass parameters to commands

Some commands accept parameters. They may be optional or mandatory and have to be identified by their names. The details of the parameters are specified in the `plugin.xml` files inside the Jar files below the `portal/plugins` directory of the Teamcenter installation directory.

In order to specify parameters for a command connected to a custom menu item you need to create the Parameters sub preference with the IDs of the parameters. Below that create the Value sub preference with the Value of the parameter. If you choose the IDs to be equal to the names of the parameters you're done. Otherwise create the Name sub preferences with the names of the parameters.

5.6.1 Example: Create a new Item

This example demonstrates how to invoke a command with a parameter through a custom menu item. The command opens the wizard to create a new Document type. The type can be revised in the dialog that opens when the user clicks on the menu item.

new_item_command_menu_item.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String" array="true"
            disabled="false" protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level custom
                menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>NewItem</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewItem.DisplayName"
            type="String" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies a display name for
                a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>Create a new Workflow Process</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewItem.Command.CommandId"
            type="String" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies the command id of
                the command to be invoked through this custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>com.teamcenter.rac.common.AddNew</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewItem.Command.Parameters"
            type="String" array="true" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference lists the parameters of the
                command to be invoked through this custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>objecttype</value>
            </context>
        </preference>
        <preference
            name="T4X.CustomMenuItems.NewItem.Command.Parameters.objecttype.Value"
            type="String" array="false" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference specifies the value of a
                command parameter.
            </preference_description>
            <context name="Teamcenter">
                <value>Document</value>
            </context>
        </preference>
    </category>
```

```
</preferences>
```

6 Preferences to specify (custom) actions

Note

- Custom actions are not supported by T4EA

6.1 Overview

Actions fall into two categories:

1. Predefined actions are those which are defined out of the box. In general the sets of actions are different for every flavor of T4x. For example T4S comes with Create, Create Direct, Change, Change Hidden, Display, etc.
2. They only appear in the Gateway menu if they belong to use cases which are configured for the GUI. Some actions normally never appear in the Gateway menu because they get invoked independently from use cases. For example the ShowCustomData action is executed whenever the user requests data from the EA system to be displayed in the Data View.

The details of these actions need to be modified only rarely. For example the admin may want to by-pass the transaction window for read-only actions.

Custom actions can be defined by the admin to provide the user with additional functionalities, for example to start a particular workflow. The admin most probably wants to adjust their details to fit the user's need.

Both types of actions share most of their attributes, so the tweaking you can apply to custom actions can in many cases also be applied to predefined actions.

Some attributes get set through preferences which are also or even exclusively evaluated by the T4x server. The names of such preferences start with `T4S_` , `T4O_` or `T4EA_` but **not** `T4X_`. In the following such attributes are given with their preference names where *MenuItemId* stands for the ID of the menu item and T4?_ for the flavor specific prefix.

6.2 Attributes common to both Types of Actions

The following attributes, given with the names of the corresponding sub preferences, are primarily intended to specify custom actions but can also be used for predefined actions:

AutoPerform : logical

If set to true the action will be executed immediately without waiting for the user to click on the transaction window's "Perform" button. The transaction window will open no matter the value of this preference. Compare with SilentPerform .

SilentPerform : logical

If set to true the action will be executed immediately completely by-passing the transaction window. Compare with AutoPerform .

RefreshTcObject : logical

If set to true the Teamcenter object on which to apply the action will be refreshed after executing the action so changes of its attributes become visible. This is useful for actions that change the state of the Teamcenter object.

RequiresEaConnection : logical

If set to true the GUI will ensure that the connection to the EA system has been established before executing the action.

ProvideFeedback : logical

If set to true a dialog window will appear upon a successfully completed action.

OpenUrl : logical

If set to true the action is supposed to return a URL which will then be opened in a HTML browser.

6.3 Preferences for Custom Actions

The following attributes, given with the names of the corresponding preferences, are evaluated by the GUI as well as the T4x server:

T4?_Gateway_Menu_Custom_*MenuItemId*_ObjectType : String

The EA object type to which this action applies e.g. "MaterialMaster"

T4?_Gateway_Menu_Custom_*MenuItemId*_ObjectDataRequired : logical

true if the selected object is to be passed with the action

T4?_Gateway_Menu_Custom_*MenuItemId*_TypeList : Array of String

The list of types of Teamcenter objects which may be passed with the action

The following attributes, given with the names of the corresponding preferences, are exclusively evaluated by the T4x server:

T4?_Gateway_Menu_Custom_*MenuItemId*_Mode : String

- If the value is "WORKFLOW" this custom action creates a new workflow process. The process is further specified by a number of preferences which correspond to text fields the user can fill out in the "New Process Dialog" dialog as depicted below:

  The preferences are as follows:

  T4?_Gateway_Menu_Custom_*MenuItemId*_JobDescription

  Corresponds to the "Description" field in the "New Process Dialog" dialog.

  T4?_Gateway_Menu_Custom_*MenuItemId*_JobName

  Corresponds to the "Process Name" field in the "New Process Dialog" dialog.

T4?_Gateway_Menu_Custom_*MenuItemId*_ProcedureName

Corresponds to the "Process Template" field in the "New Process Dialog" dialog

- If the value is not "WORKFLOW" this custom action causes the invocation of a TCL procedure which must be defined in a `.sd` file below the `var/mmap` directory inside the T4x installation directory. The procedure receives the single parameter `TransactionId`. It is further specified by the following preferences:

  T4?_Gateway_Menu_Custom_*MenuItemId*_ObjectCustomCall

  Fully qualified name of the procedure to invoke.

More data can be obtained through the global `::TcData` array like in regular mapping definitions.

6.4 Common Examples

For the Icon sub preference we copied the newprocess.png file out of Teamcenter to the directory where the Teamcenter Rich Client is installed. The following picture will then appear on the left side of the menu item's name:

6.4.1 Example: Create a new Workflow Process through a custom action

This example demonstrates how to create a new workflow process through a custom action. The custom action calls into the T4x server which performs the actual task of creating the process.

This example configures T4x to create the new process from the "T4S_MM" template shipped with T4S. Note that T4x requires the process name and description to be set although these don't have to be set in the "New Process Dialog" dialog.

new_process_custom_action.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String"
            array="true" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level
                custom menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>NewT4SMMProcess</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewT4SMMProcess.DisplayName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies a display
                name for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>New T4S_MM Workflow Process</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.NewT4SMMProcess.Icon"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the path
                of an icon for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>${env.TPR}/newprocess.png</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_Mode"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies whether
                this custom action is supposed to create a new workflow
                process or to invoke a TCL procedure. In the first case
                the value would be WORKFLOW. Anything else would cause a
                TCL procedure to be invoked.
            </preference_description>
            <context name="Teamcenter">
                <value>WORKFLOW</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_TypeList"
            type="String" array="true" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the list
                of types of Teamcenter objects which may be passed with
                the action.
```

```
            </preference_description>
            <context name="Teamcenter">
                <value>SAP2_T4S_Item Revision</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_ObjectType"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the EA
                object type to which this action applies e.g.
                "MaterialMaster".
            </preference_description>
            <context name="Teamcenter">
                <value>MaterialMaster</value>
            </context>
        </preference>
        <preference
            name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_ProcedureName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the name
                of the workflow template from which to instantiate the
                new process. It corresponds to the "Process Template"
                field in the "New Process" dialog.
            </preference_description>
            <context name="Teamcenter">
                <value>T4S_MM</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_JobName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the name
                of the workflow process to be created. It corresponds to
                the "Process Name" field in the "New Process" dialog.
            </preference_description>
            <context name="Teamcenter">
                <value>T4S MM through custom action</value>
            </context>
        </preference>
        <preference
            name="T4S_Gateway_Menu_Custom_NewT4SMMProcess_JobDescription"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description> This preference specifies the
                description of the workflow process to be created. It
                corresponds to the "Description" field in the "New
                Process" dialog.
            </preference_description>
            <context name="Teamcenter">
                <value>T4S MM through custom action</value>
            </context>
        </preference>
    </category>
</preferences>
```

6.4.2 Example: call a TCL procedure through a custom action

This example demonstrates how to call a TCL procedure defined in a .sd file in the `var/mmap` directory. You can try it with T4S. For the other T4x flavors just replace the T4S-specific prefixes with those of the desired flavor. The example writes the contents of the global `::TcData` array to the session log file.

You need to perform some extra steps to make the example running:

1. Create the t4s_custom_actions.sd file inside the var/mmap directory and copy&paste the following lines into it:

```
namespace eval ::T4S::CUSTOM::MAPPING {
  namespace export logTcData

  proc logTcData {TransactionId args} {
    log "TransactionId = $TransactionId"
    log "Contents of the TcData array:"
    set keys [lsort -dictionary [array names ::TcData]]
    foreach key $keys {
      log "  \$::TcData($key) = \"$::TcData($key)\""
    }
    set ::StatusInfo(ReverseMappingStatus) \
      [::T4S::MM::MAPPING::SAP_MaterialMaster2TC_Object \
            $TransactionId SKIPPED {}]
    return SKIPPED
  }

  proc log {message} {
    tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] \
      -mtype INTERN $message"
  }

}
```

2. Add the following line to the t4s_mapping_config.sd file:

```
source -relax t4s_custom_actions.sd
```

3. Run the mmap tool
4. Copy the t4s_mapping_config.rfdt file from the tmp to the lib directory.
5. Restart T4x
6. Instead of the steps 3-5 it is much for efficient to do the “mapping hot deploy”, see chapter “How to modify the mapping” in the documentation “Configuration Guide” of the corresponding T4x product

So here is the preference file:

procedure_custom_action.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.CustomMenuItems" type="String"
            array="true" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lists the top level
                custom menu items.
            </preference_description>
            <context name="Teamcenter">
                <value>:separator</value>
                <value>LogTcData</value>
            </context>
        </preference>
        <preference name="T4X.CustomMenuItems.LogTcData.DisplayName"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies a display
                name for a custom menu item.
            </preference_description>
            <context name="Teamcenter">
                <value>Log TcData Array</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_LogTcData_Mode"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies whether
                this custom action is supposed to create a new workflow
                process or to invoke a TCL procedure. In the first case
                the value would be WORKFLOW. Anything else would cause a
                TCL procedure to be invoked.
            </preference_description>
            <context name="Teamcenter">
                <value>INTERACTIVE</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_LogTcData_TypeList"
            type="String" array="true" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the list
                of types of Teamcenter objects which may be passed with
                the action.
            </preference_description>
            <context name="Teamcenter">
                <value>SAP2_T4S_Item Revision</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_LogTcData_ObjectType"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
```

```
            <preference_description>This preference specifies the EA
                object type to which this action applies e.g.
                "MaterialMaster".
            </preference_description>
            <context name="Teamcenter">
                <value>MaterialMaster</value>
            </context>
        </preference>
        <preference name="T4S_Gateway_Menu_Custom_LogTcData_ObjectCustomCall"
            type="String" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference specifies the fully
                qualified name of the procedure to invoke.
            </preference_description>
            <context name="Teamcenter">
                <value>::T4S::CUSTOM::MAPPING::logTcData</value>
            </context>
        </preference>
    </category>
</preferences>
```

6.4.3 Example: Suppress Log-On dialog in the data view context

This example demonstrates how to avoid a log-on dialog to pop up before fetching data from an EA system to be presented in the data view. Note that the connection to the EA system must be established by other means, for example by the mapping procedures. Adapt the example if you need it for other actions.

An alternate method may be to set up auto-log-in using the command `::T4S::CONNECTION2SAP::setConnectionInfoPlain2` from within the mapping config.

suppress_log-on.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference
            name="T4X.Actions.ShowCustomData.RequiresEaConnection"
            type="Logical" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference avoids a log-on
                dialog being opened before executing the action.
            </preference_description>
            <context name="Teamcenter">
                <value>false</value>
            </context>
        </preference>
    </category>
</preferences>
```

6.4.4 Example: Bypass Transaction Window for read-only Actions

This example demonstrates how to let read-only actions execute immediately by-passing the transaction window. This is not applicable to T4EA as it does not have a transaction window.

by-pass_tx_window.xml

```
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.Actions.Display.SilentPerform"
            type="Logical" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lets this action
                execute immediately by-passing the transaction window.
            </preference_description>
            <context name="Teamcenter">
                <value>true</value>
            </context>
        </preference>
        <preference name="T4X.Actions.HttpDisplay.SilentPerform"
            type="Logical" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lets this action
                execute immediately by-passing the transaction window.
            </preference_description>
            <context name="Teamcenter">
                <value>true</value>
            </context>
        </preference>
        <preference name="T4X.Actions.ProductStructure.SilentPerform"
            type="Logical" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lets this action
                execute immediately by-passing the transaction window.
            </preference_description>
            <context name="Teamcenter">
                <value>true</value>
            </context>
        </preference>
        <preference name="T4X.Actions.HttpProductStructure.SilentPerform"
            type="Logical" array="false" disabled="false"
            protectionScope="Site" envEnabled="false">
            <preference_description>This preference lets this action
                execute immediately by-passing the transaction window.
            </preference_description>
            <context name="Teamcenter">
                <value>true</value>
            </context>
        </preference>
    </category>
</preferences>
```

7 Dealing with use cases

Use cases combine actions and EA object types, e.g. DisplayMaterial where Display denotes the action and Material the EA object type.

7.1 Example: Suppress EA object types in the data view

When setting up T4x the admin assigns EA object types to Teamcenter object types through preferences whose names contain the names of the EA object types and whose values the names of the Teamcenter object types. Their names start with the name of the flavor and end with `TypeList`. For example the `T4S_MaterialMasterTypeList` preference can contain the values CommercialPart Revision, ManufacturerPart Revision or names of business objects defined by the customer. They define which Teamcenter object can be transferred to which EA object. In the following we will call them TypeList preferences.

As mentioned in section *2.6 Compatibility layer* on page 10 the current GUI uses a different naming scheme. Moreover it allows for a more fine-grained use case based assignment between EA and TC object types that also includes the action. By default the GUI uses the old naming scheme. It is however possible to override preferences following the old scheme by ones following the new scheme.

The ID of the action that fetches and displays data from EA systems is `ShowCustomData`. The ID of a use case results from appending the ID of an EA object type. So `ShowCustomDataMaterial` would be the ID of the “show the data of an SAP MaterialMaster” use case.

Use cases have a `SupportedTcObjectTypes` attribute to determine to which types of Teamcenter object they are applicable.

In order to avoid EA object types appearing in the data view’s EA object type chooser you would set the `T4X.UseCases.ShowCustomData`*`EaObjectTypeID`*`.SupportedTcObjectTypes` preference to a value that does not identify a Teamcenter object type, e.g. `NONE`, replacing *`EaObjectTypeID`* with then ID of an EA object type.

The following example removes the `Change` and `Equipment` types from the EA object type chooser:

suppress_target_types_for_data_view.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UseCases.ShowCustomDataChange.SupportedTcObjectTypes"
            type="String" protectionScope="Site" array="true" disabled="false"
            envEnabled="false">
            <preference_description>
                This preference specifies the Teamcenter
                types that are supported for the data view when it shows data
                for the Change target type.
            </preference_description>
            <context name="Teamcenter">
                <value>NONE</value>
            </context>
        </preference>
        <preference
            name="T4X.UseCases.ShowCustomDataEquipment.SupportedTcObjectTypes"
            type="String" protectionScope="Site" array="true" disabled="false"
            envEnabled="false">
            <preference_description>
                This preference specifies the Teamcenter
                types that are supported for the data view when it shows data
                for the Equipment target type.
            </preference_description>
            <context name="Teamcenter">
                <value>NONE</value>
            </context>
        </preference>
    </category>
</preferences>
```

8 Preferences affecting the UI's appearance

The preferences whose names start with `T4X.UI.` can be used to modify the appearance of the RAC GUI. The preferences dealing with the Gateway menu are in this namespace as well but are covered in a separate chapter. See chapter *3 Preferences covering the Gateway menu* on page 11.

8.1 Example: Define the tabs shown in the T4x transaction window

Note

- The transaction window is not supported by T4EA

The T4x transaction window opens whenever a transaction is started from the T4x Gateway menu. By default it shows three tabs. Their presence and order can be modified through the `T4X.UI.Transaction.Tabs` preference. It is an array preference whose values can be any of the following:

Attributes

shows the Teamcenter data based on a given style sheet

CustomData

shows the ERP Data view (e.g. for T4S: the "SAP Data view")

Status

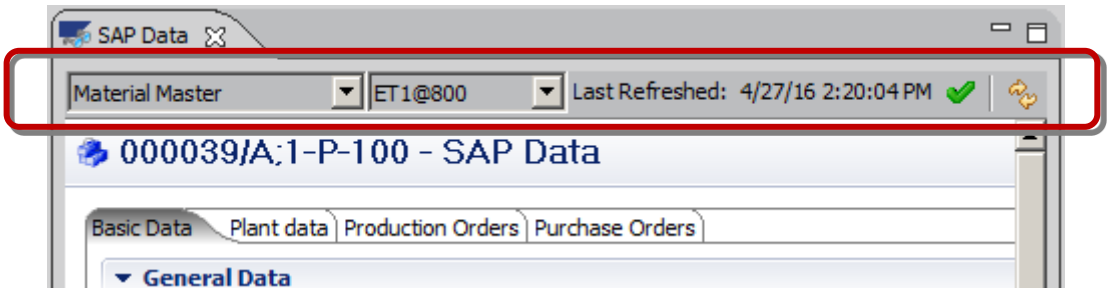
shows the detailed status messages, i.e. especially the last error message

tabs_in_transaction_window.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.Transaction.Tabs" type="String"
            protectionScope="Site" array="true" disabled="false" envEnabled="false">
            <preference_description> This preference specifies the presence and
                order of the tabs in the transaction window of the T4x RAC UI
            </preference_description>
            <context name="Teamcenter">
                <value>Status</value>
                <value>Attributes</value>
            </context>
        </preference>
    </category>
</preferences>
```

8.2 Example: Define the elements shown in the Data View's control panel

At the top of the T4x Data View you find a number of widgets that help users to control the data view as shown in the figure below:



Their presence and order can be modified through the `T4X.UI.CustomData.ControlWidgets` preference. It is an array preference whose values are listed below. The list also shows the default value of the preference.

EaConnection

A chooser widget allowing the user to choose the EA system to connect with

EaObjectType

A chooser widget allowing the user to choose the type of the EA object (= target type). The EA object is associated with the selected Teamcenter object and typically the result of a data transfer.

Separator

A vertical line used to visually separate the control widgets from each other.

LastRefreshed

A field showing date and time of the last refresh operation.

Progress

A progress indicator. It is an icon showing whether a refresh operation is in progress, has finished with success or with an error.

Separator

See above.

Refresh

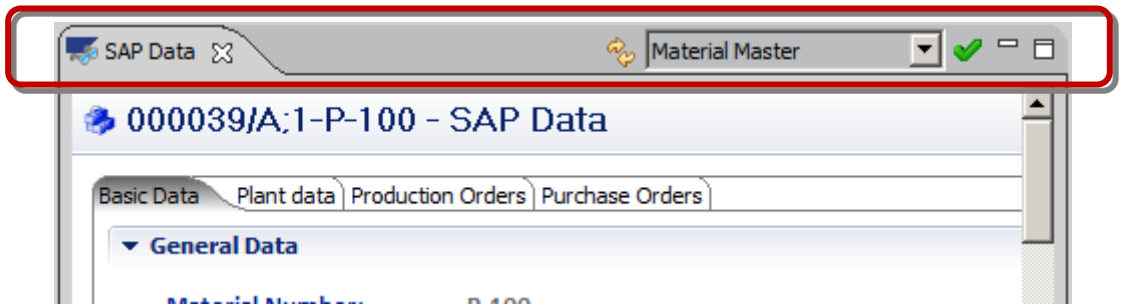
A button the user can click to initiate a refresh operation.

Look at the example preference definition:

data_view_control_panel.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.CustomData.ControlWidgets" type="String"
            protectionScope="Site" array="true" disabled="false" envEnabled="false">
            <preference_description>This preference specifies the presence and
                order of the widgets in the data view's control panel
            </preference_description>
            <context name="Teamcenter">
                <value>Refresh</value>
                <value>EaObjectType</value>
                <value>Progress</value>
            </context>
        </preference>
    </category>
</preferences>
```

This example reduces the control panel to contain only the refresh button the EA object type chooser and the progress indicator as shown in the figure below. Compare this with the figure at the beginning of this section. Note how the position of the control panel has changed due to its smaller width. It has moved from the view to the tab header leaving more vertical space for the EA data:



If T4x is configured to always use a particular connection it makes sense to leave off the connection chooser. The connection can be defined in the mapping configuration through the `selectActiveConnection2SAP`, `selectActiveConnection2EBS` or `selectActiveConnection2EA` resp.

8.3 Example: Define the height of the T4x connections window

Depending on the T4x flavor clicking on **SAP Connections…** , **EBS Connections…** or **EA Connections…** resp. form the Gateway menu opens the T4x connections window.

In order to control its height you may change the `T4X.UI.EaConnections.Height` preference. It defaults to 200 pixel which provide enough space for four entries without having to scroll down.

change_dimensions.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.EaConnections.Height" type="String"
            protectionScope="Site" array="false" disabled="false" envEnabled="false">
            <preference_description> This preference specifies the height of the
                Connections Window of the RAC UI
            </preference_description>
            <context name="Teamcenter">
                <value>300</value>
            </context>
        </preference>
    </category>
</preferences>
```

If you replace *Height* with *Width* you can modify the initial width of a window or dialog.

In order to modify the dimensions of other types of UI elements you can replace *EaConnections* with their IDs as listed below:

EaConnections

The connection window as shown in the example

Transaction

The transaction window (not available in T4EA)

Messages

The message popup window

ShowLog.InternalBrowser

The browser window shown when **Show Log** is selected from the Gateway menu.

ShowWebPage.InternalBrowser

The browser window shown for actions that return URLs or for hyperlink elements placed in the data view.

8.4 Example: Define when to collapse message details

Error messages often have a details part providing information normally relevant only for the expert user. Message boxes display the message details in an area titled **Details**. It can be in a collapsed or expanded state. The user can change the state by clicking on the title.

By default message details with up to ten lines are immediately shown. When they have more lines they are hidden and the user must click on the title to make them visible.

The threshold value can be changed through the `T4S.UI.Messages.Details.ExpandedMaxLines` preference. If the value is 0 the details are

initially hidden no matter their number of lines. The value -1stands for infinity meaning that the details are initially visible no matter the number of lines. As mentioned the default is ten.

The below example sets the value to 0 meaning that the details will be initially hidden.

message_details_appearance.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.Messages.Details.ExpandedMaxLines"
            type="Integer" protectionScope="Site" array="false" disabled="false"
            envEnabled="false">
            <preference_description>This preference specifies how the details
                part of message boxes of the T4x RAC UI are displayed
            </preference_description>
            <context name="Teamcenter">
                <value>0</value>
            </context>
        </preference>
    </category>
</preferences>
```

8.5 Example: Change the labels of UI elements

The labels of many UI elements of the T4x RAC GUI can be changed through Teamcenter preferences. In order to get the names of these preferences inspect the `$TC_ROOT/portal/plugins/de.tesis.plmware.t4x.app.jar` file with a Zip utility. You may have to create a copy and change the name to end with `.zip`. Inside that file you find the `properties` folder that contains localized properties files, e.g. `properties/texts_de.properties`. Use these files to identify the property that you want to override. The names of the preferences can be derived from the names of the properties by prefixing them with `T4X.`, `T4S.`, `T4O.` or `T4EA.`

The example below changes the name of the Gateway menu from "T4S Gateway" to "SAP", . "T4O Gateway" to "EBS" or "T4EA Gateway" to "EA"

Note that `${appData.eaShortName}` acts as a placeholder that expands to "SAP" in case of T4S, to "EBS" in case of T4O or to "EA" in case of T4EA. You can also use placeholders to provide language specific labels. See section *2.3 Template preferences* on page 9 and the following.

change_labels.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="10.0">
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4X.UI.GatewayMenu.Label" type="String"
            protectionScope="Site" array="false" disabled="false" envEnabled="false">
            <preference_description>This preference specifies the label of the
                Gateway menu of the T4x RAC UI
            </preference_description>
            <context name="Teamcenter">
                <value>${appData.eaShortName}</value>
            </context>
        </preference>
    </category>
</preferences>
```

9 Control by which style sheet the attributes tab is rendered

Note

- This does not apply to T4EA because it does not have a transaction window

9.1 Introduction

The presentation of the Attributes tab in the Transaction window is determined by so called style sheets. A style sheet is a data set of XMLRenderingStylesheet type with a single XML file. For their format please consult the documentation shipped with Teamcenter. The names of the XML files end with .xml in contrast to the names of the data sets.

Caution

- When importing files into data sets Teamcenter suggests using their names for the data sets. Take care to remove the .xml file name extensions.

9.2 List the style sheets you want to use

You start configuring the Attributes tab by declaring a list of IDs each of which represents a *stylesheet declaration*. You add the IDs of the stylesheet declarations to the preference `T4X.AttributesStylesheets` (maybe substitute T4X. with a flavor-specific prefix).

Each stylesheet declaration points to a single stylesheet. Multiple stylesheet declarations may point to the same style sheet.

You would normally choose the name of the data set for the ID in order to let a style sheet declaration point to a style sheet. If there is a requirement to give them different values create the `Rendering` sub preference and set it to the name of the data set.

The style sheet declarations listed in the `T4X.AttributesStylesheets` preference are the candidates from which the T4x GUI selects one by matching them with the object selected under **Objects** in the transaction window. If you finished at this point it would select the first one from the list no matter the selected object. In the following sections you will learn how to trim the style sheets to let the Attributes tab becoming aware of the selected object.

9.3 The Algorithm

You first need to know that the selected object represents a transaction (also known as "transfer") and as such it has the following attributes:

EA object type (also known as target type)

The type of the object in the EA system to which the transaction is applied

EA connection

The Connection to use for the transaction

TC object type

The type of the object in Teamcenter to which the transaction is applied, e.g. Design Revision

Type Hierarchy

A list containing the Teamcenter object type and all its super types, starting with the Teamcenter object type

The principle by which the GUI selects the most suitable style sheet is then pretty simple:

It starts by creating a list of style sheet declarations which gets initialized with the content of the `T4X.AttributesStylesheets` preference. We will call this the *candidates list*.

It then walks through the list of candidates in the order given through the preference and throws out those ones that are not suitable with respect to a particular transaction attribute, for example the type of the selected object. In other words it removes style sheet declarations that have been trimmed to require a particular transaction attribute and that requirement is not fulfilled.

It repeats the step above but this time taking another attribute into account.

Filtering with all attributes eventually results in a new candidates list. If it is empty a built-in default style sheet will be used. Otherwise the first one determines the rendering.

9.4 Add sub preferences to specialize your style sheets

The following sub preferences can be used to trim your style sheet declarations:

EaObjectTypes

This array sub preference must contain the EA object type (also known as "target type") of the selected transaction for the style sheet declaration to remain in the candidates list. If not given or empty all EA object types will be accepted.

EaConnections

This array sub preference must contain the EA connection of the selected transaction for the style sheet declaration to remain in the candidates list. If not given or empty all EA connections will be accepted.

Types

This array sub preference must contain one of the types from the type hierarchy of the selected transaction for the style sheet declaration to remain in the candidates list. If not given or empty all Teamcenter object types will be accepted.

The T4x GUI starts with the first type of the type hierarchy and goes through the entire candidates list until it finds a style sheet declaration whose Types attribute contains that type. If it finds one, that will be used for the rendering. Otherwise the GUI continues with the second type of the type hierarchy and so on.

This way you can specialize a style sheet for particular type (e.g. Design Revision) and have a fallback style sheet for its super types (e.g. ItemRevision).

Example:

This example is inspired by the set of preferences shipped with Teamcenter Gateway *for SAP Business Suite* (T4S). Note that it contains two style sheet declarations for SAP documents, one for DataSets and another one for ItemRevisions:

stylesheet_declarations.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences>
    <category name="SAP Gateway">
        <category_description></category_description>
        <preference name="T4S.AttributesStylesheets" type="String"
            array="true" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference defines the list of
                style sheets available for the T4S transaction window
            </preference_description>
            <context name="Teamcenter">
                <value>T4SStylesheetMaterial</value>
                <value>T4SStylesheetDocument4Dataset</value>
                <value>T4SStylesheetDocument4ItemRevision
                </value>
            </context>
        </preference>
        <preference
     name="T4S.AttributesStylesheets.T4SStylesheetMaterial.EaObjectTypes"
           type="String" array="true" disabled="false" protectionScope="Site"
            envEnabled="false">
            <preference_description>This preference assigns a given style sheet
               (defined by the corresponding T4S.AttributesStylesheets preference)
                to a defined transfer target of the target system
            </preference_description>
            <context name="Teamcenter">
                <value>Material</value>
            </context>
        </preference>
        <preference
```

```
      name="T4S.AttributesStylesheets.T4SStylesheetDocument4Dataset.EaObjectTypes"
             type="String" array="true" disabled="false" protectionScope="Site"
             envEnabled="false">
             <preference_description>This preference assigns a given
                 style sheet (defined by the corresponding
                 T4S.AttributesStylesheets preference) to a defined
                 transfer target of the target system
             </preference_description>
             <context name="Teamcenter">
                 <value>Document</value>
             </context>
         </preference>
         <preference
      name="T4S.AttributesStylesheets.T4SStylesheetDocument4Dataset.Types"
             type="String" array="true" disabled="false" protectionScope="Site"
             envEnabled="false">
             <preference_description>The preference assigns a given style
                 sheet (defined by the corresponding
                 T4S.AttributesStylesheets preference) to a defined
                 transfer source of the source system
             </preference_description>
             <context name="Teamcenter">
                 <value>DirectModel</value>
                 <value>UGMASTER</value>
                 <value>MSWord</value>
             </context>
         </preference>
         <preference
      name="T4S.AttributesStylesheets.T4SStylesheetDocument4ItemRevision.EaObjectTypes"
             type="String" array="true" disabled="false" protectionScope="Site"
             envEnabled="false">
             <preference_description>This preference assigns a given
                 style sheet (defined by the corresponding
                 T4S.AttributesStylesheets preference) to a defined
                 transfer target of the target system
             </preference_description>
             <context name="Teamcenter">
                 <value>Document</value>
             </context>
         </preference>
         <preference
             name="T4S.AttributesStylesheets.T4SStylesheetDocument4ItemRevision.Types"
             type="String" array="true" disabled="false" protectionScope="Site"
             envEnabled="false">
             <preference_description>The preference assigns a given style
                 sheet (defined by the corresponding
                 T4S.AttributesStylesheets preference) to a defined
                 transfer source of the source system
             </preference_description>
             <context name="Teamcenter">
                 <value>SAP2_T4S_Item Revision</value>
             </context>
         </preference>
     </category>
</preferences>
```

10 File level customizations

All the customizations described in the previous chapters could be achieved by creating and modifying Teamcenter preferences. This is possible because Java bytecode contained in the plug-in files is executed by the Java Virtual Machine (JVM). The bytecode reads the values of the preferences and modifies the user interface accordingly. Other types of customizations happen before plug-ins get loaded and their bytecode being executed.

These types of customizations are described in this chapter. They require you to open up plug-in jar files, change files inside them and close them. If you do not have an editor that can edit files inside a Jar file extract their contents into a directory using a Zip extraction tool, edit the files in the directory and create a new Jar file from the directory. You may have to rename the Jar file so that its name ends with `.zip` and vice versa.

Caution

- Be careful to maintain the directory hierarchy inside the Jar file. Verify that you find a file named `plugin.xml` and a directory named `META-INF` at its root.

10.1 Example: Add T4S data view to My Teamcenter

Teamcenter applications such as **My Teamcenter** or **Structure Manager** are in fact Eclipse perspectives. Perspectives describe the sets of tools available in the application and how they are laid out.

Perspectives are identified by IDs. For example the My Teamcenter perspective has the ID `com.teamcenter.rac.ui.perspectives.navigatorPerspective` and the Structure Manager `com.teamcenter.rac.pse.PSEPerspective`.

Perspectives can be saved in Teamcenter under **Window → Save Perspective As** but the saved Perspectives cannot easily be distributed and get lost when the rich client's cache directory under `%PROFILEUSER%\Teamcenter\RAC` is removed.

In order to add a view to a perspective a plug-in's plugin.xml file has to be changed. Plug-ins are files whose names end with `.jar`. You find them under `%TC_ROOT%/portal/plugins`.

For T4S edit the `plugin.xml` file inside the `de.tesis.plmware.t4s.app.jar` file. Add the following snippet:

plugin.xml

```
<extension point="org.eclipse.ui.perspectiveExtensions">
     <perspectiveExtension
         targetID="com.teamcenter.rac.ui.perspectives.navigatorPerspective">
         <view id="de.tesis.plmware.t4s.app.views.eadata"
             relative="com.teamcenter.rac.ui.views.ViewerView"
             relationship="bottom"
             ratio="0.50" />
     </perspectiveExtension>
</extension>
```

This will cause the data view to appear below the area containing the **Summary**, **Details**, **Viewer** etc. tabs.

Go to **https://help.eclipse.org** and enter `org.eclipse.ui.perspectiveExtensions` into the search field to learn more.

Siemens Industry Software

Headquarters

Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas

Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe

Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific

Suites 4301-4302, 43/F
AIA Kowloon Tower, Landmark East
100 How Ming Street
Kwun Tong, Kowloon
Hong Kong
+852 2230 3308

About Siemens PLM Software

Siemens PLM Software, a business unit of the Siemens Industry Automation Division, is a leading global provider of product lifecycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide.
Headquartered in Plano, Texas, Siemens PLM Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens PLM Software products and services, visit www.siemens.com/plm.