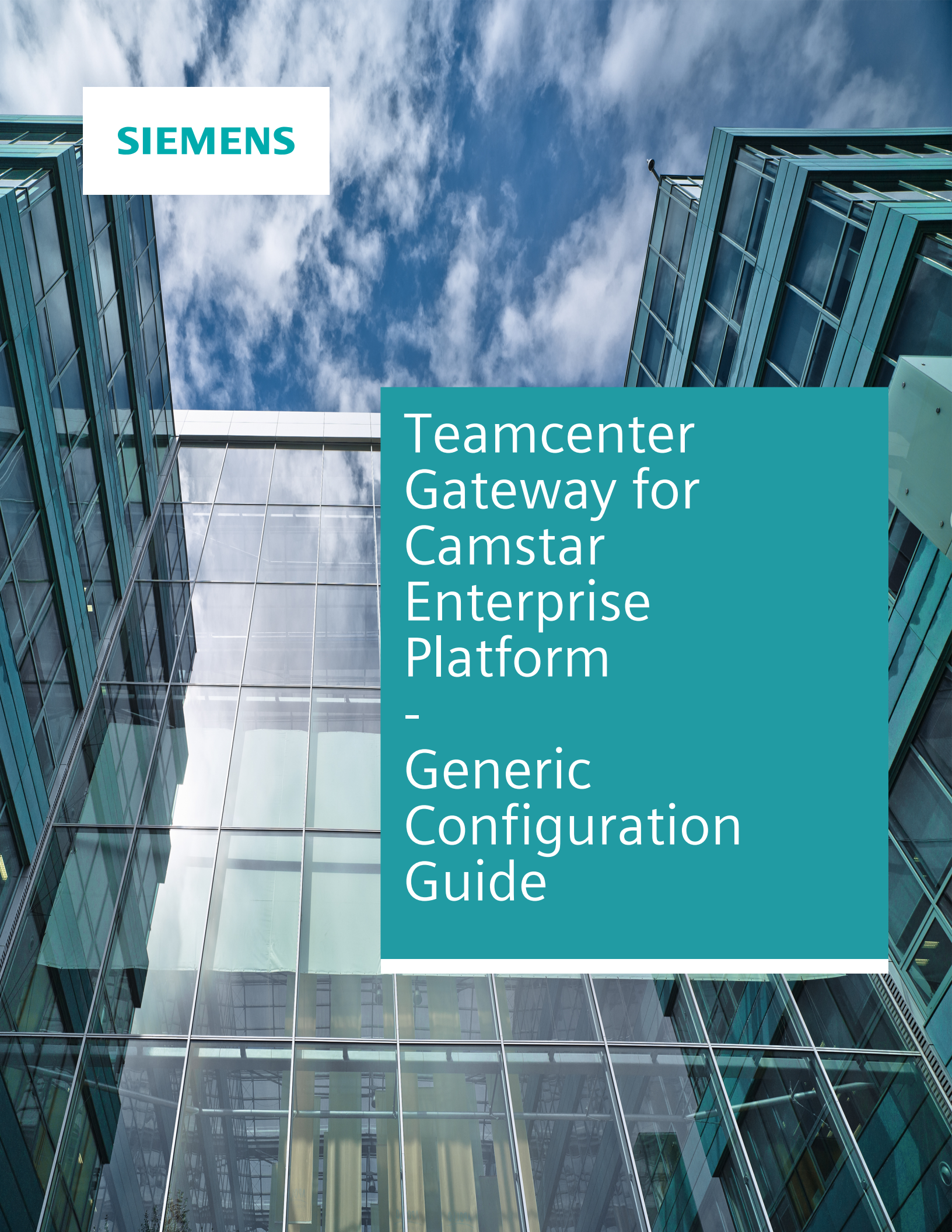


SIEMENS

Teamcenter Gateway for Camstar Enterprise Platform - Generic Configuration Guide

Contents

Preface

This documentation cannot be used as a substitute for consulting advice, because it can never consider the individual business processes and configuration. Despite our best efforts it is probable that some information about functionality and coherence may be incomplete.

Issue: August 2019

1. Introduction

The Teamcenter Gateway software solutions are general purpose integration software that provide data and process integration between Teamcenter by Siemens Product Lifecycle Management Software Inc. and particular target system you want to integrate with.

Please note that the Teamcenter Gateway for SAP Business Suite (T4S), the Teamcenter Gateway for SAP S/4HANA (T4S4), the Teamcenter Gateway for Oracle EBS (T4O), the Teamcenter Gateway for Camstar Enterprise Platform (T4CEP), the SIMATIC IT UA DM Gateway for Teamcenter (CLM4T) and the Teamcenter Gateway for Enterprise Applications (T4EA) are available as dedicated products to support integration with the SAP Business Suite®, the Oracle E-Business Suite, the Camstar Enterprise Platform, SIMATIC IT UA DM and other database enterprise systems, respectively. Each of these products can be used as a single installation or as a combined installation. All of this products provides a wide range of workflow functions to transfer and synchronize data between Teamcenter and Enterprise Applications.

The main functions cover the following functional areas:

- Unstructured Teamcenter types or objects (item revisions, datasets)
- Structured Teamcenter types or structures (bill of materials)

You can adapt the individual functions to your requirements by applying specific modifications to the data mapping process. The steps are described in detail in this manual or the product-specific manuals.

For more information on new components and new versions of the products visit

https://www.plm.automation.siemens.com/en/products/active-integration/index.shtml

2. Concept of Triggering Transfer

T4x is a bidirectional gateway to exchange data between Teamcenter and EA so that you can start the transfer from Teamcenter or the EA. The way to trigger the data transmission from Teamcenter can be divided into the following groups:

- **process-based**
- **external triggers**

2.1 Process Based Trigger (Workflow)

A workflow is the automation of business procedures in which documents, information, or tasks are passed from one participant to another in a way that is governed by rules or procedures. Teamcenter workflows allow you to manage your product data processes. You can create any type of workflow to accommodate your business procedures. In T4x context, the workflows are a powerful mechanism to trigger the transfer or import process.

Handlers are the lowest-level building blocks in the workflow. They are small ITK programs used to extend and customize tasks. There are two kinds of handlers:

- Action handlers extend and customize task actions. They perform such actions as displaying information, retrieving the results of previous tasks (inherit), notifying users, setting object protections and launching applications.
- Rule handlers integrate workflow business rules into EPM workflow processes at the task level. They attach conditions to an action. Rule handlers confirm that a defined rule has been satisfied. If the rule is met, the handler returns the `EPM_go` command, allowing the task to continue. If the rule is not met, it returns the `EPM_nogo` command, preventing the task from continuing. Many conditions defined by a rule handler are binary (that is, they are either true or false). However, some conditions are neither true nor false. EPM allows two or more rule handlers to be combined using logical AND/OR conditions. When several rule handlers are combined using a logical Or condition, rule handler quorums specify the number of rule handlers that must return go for the action to complete.

T4x provides this kind of functions for supporting workflows. Please see **T4x Workflow Handler** to learn more about this workflow handlers.

Please also check this chapters for additional information:

- **Continue with Workflow Job despite of a T4x Error**
- **Attach Additional Teamcenter Objects to a Workflow and Read their Attributes**
- **Attach a Teamcenter form to the current workflow job and read it via the Teamcenter Gateway**

- **Attach more functionality to a workflow**
- **Obtain Detailed Information about the Workflow**

2.1.1 Continue with Workflow Job despite of a T4x Error

By default, a workflow job is stopped when an error occurs. In order to prevent that, add the handler argument `-continue_on_error` with the value `true`. If this is specified in a T4x handler, it will ignore all Teamcenter Gateway related errors and continue without any error message so that the workflow job will run through completely and the Teamcenter Gateway errors can be handled separately. The handling of Teamcenter errors (such as "access denied" etc.) is not affected. The feature is available in all standard transfer action handlers.

For telling T4x to not stop until all targets are processed, you can use the handler argument `-collect_all_errors`.

2.1.2 Attach Additional Teamcenter Objects to a Workflow and Read their Attributes

The mechanism of **Attach a Teamcenter form to the current workflow job and read it via the Teamcenter Gateway** can also be used to attach several objects other than Forms to the workflow. Although such objects cannot be displayed in the workflow, it may be useful to read their properties in the mapping. You can add objects (Items, Revisions) that have a GRM relation to any target object of the workflow.

The solution consists of the following steps:

- Attach the desired related objects to the workflow: add one or more action handlers `EPM_attach_related_objects` to a workflow task (the root task or the task that contains the T4x transfer handler). The handlers must have these parameters for a sample case where the target has an `IMAN_specification` relation to the desired Item: `-relation=IMAN_specification, -att_type=reference, -type=ItemRevision`. If you need several objects, you can use either several `EPM_attach_related_objects` handlers or the LOV feature of this handler, see the Teamcenter documentation.

- Extend the T4x Transfer handler (e.g. `<T4x>-transfer-BillOfMaterial`) in your workflow with argument `-AddObject4Mapping` and read the desired object attributes in the TCL mapping. For detailed description please see **T4x Workflow Handler**.

2.1.3 Attach a Teamcenter form to the current workflow job and read it via the Teamcenter Gateway

Best practice: add a new standard task (not a Do-Task) and fill it in the following way.

Note that:

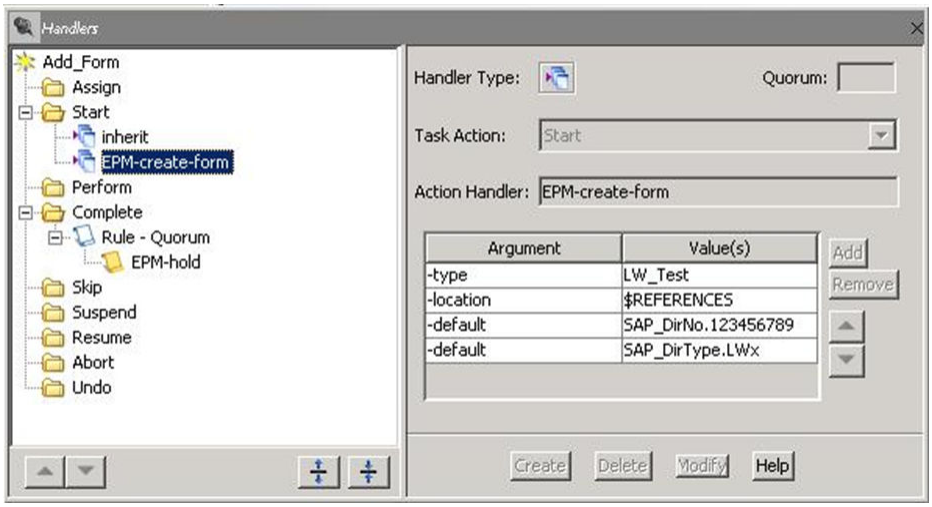
- Do not leave out the `EPM-display-form`. Then it will not be possible to switch to the Task View in the Inbox and therefore the workflow job will not be able to terminate correctly.
- Do not leave the `EPM-hold` out. Then the workflow job will break with the error `"Instance not locked`.
- The only mandatory parameter from the screen shot below is the `type`. The argument should also be used for the handler `EPM-display-form`.
- The new form can be saved e.g. as References of the current workflow job. For more possible values, see the Teamcenter help, e.g. `$TARGET`.
- You can specify initial values for each attribute with the parameter `-default`.
- For details, see Teamcenter help for action handler.

The handler `EPM-display-form` is for displaying the form in the workflow before the `Done` is activated in the Inbox.

To read the Teamcenter form attributes of the current workflow job in the TCL mapping add parameter `-AddObject4Mapping` to the handler that should read the form.

For detailed description please see **T4x Workflow Handler**.

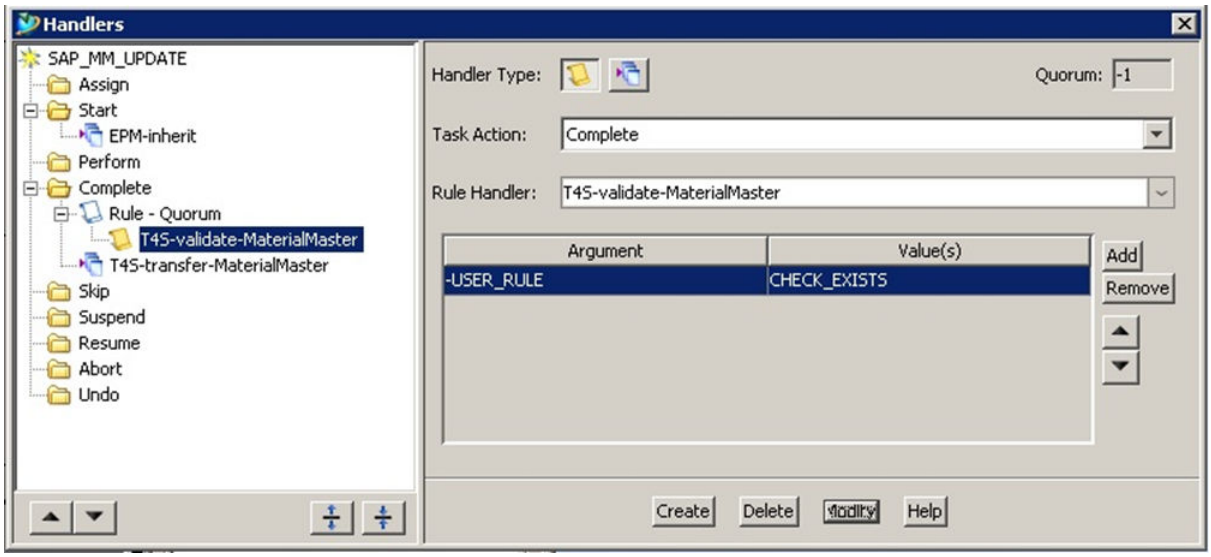
Example for the form creation:



2.1.4 Attach more functionality to a workflow

The T4x behavior can be changed by setting a Workflow argument that is evaluated in the T4x mapping.

A similar approach is evaluating a Teamcenter Workflow rule handler for setting the result `EPM_go` or `EPM_nogo`, i.e. let the Workflow job go on or stop it at that point. This is done using the T4x mapping function `callCustomerRuleHandler`. It is included in each of the object specific template mapping files (i.e. MM, DIR etc., but not LOV...) and is executed in the workflow only if the corresponding object based Rule Handler (see below) contains the argument `-user_rule` (not case sensitive; be aware of the hyphen). Of course this means that it is only executed in the workflow, not from any interactive function. The mapping function checks this argument value and sets the `Decision` accordingly. For example the Material Master mapping template may check the values `CHECK_DESCRIPTION` any `CHECK_EXISTS`. To activate `CHECK_EXISTS`, set the parameter according to the following screen shot.



- T4S will call the function `callCustomerRuleHandler` in the SAP object based rule handlers only (e.g. `T4S-validate-DocumentInfoRecord`, `T4S-validate-MaterialMaster` ...). Any rule handler that works without a specific SAP data object (e.g. `T4S-validate-SAPLogon`) will not call this function.

- The function `callCustomerRuleHandler` that is actually called is always that from the data object the current rule handler belongs to, i.e. you may configure this function for MM handling and for BOM handling differently by modifying the function in the corresponding mapping file.

- Before the function `callCustomerRuleHandler` is actually executed, the corresponding data mapping is done, so in `callCustomerRuleHandler` you may evaluate any mapping data you have set for this object.

- The behavior of the function `callCustomerRuleHandler` depends on the return value of the mapping function (e.g. for MM mapping: `TC_Object2SAP_MaterialMaster`).

- Only if this returns `OK`, the function `callCustomerRuleHandler` is called at all.

- If it returns `SKIPPED`, the Workflow Job will just go on as it immediately gets the value `EPM_go` without executing `callCustomerRuleHandler`.

- If the mapping return value is `ERROR`, the Workflow Job is stopped with the stored error message.

So if you define an `ERROR` condition in the mapping and you want to react on that without stopping the workflow job, you have to store a global variable for your error and exit the mapping function with `OK` nevertheless; then you can evaluate your variable in `callCustomerRuleHandler`.

For example, to include the `T4S-validate-ChangeNumber` (which checks a SAP ECM) in the MM workflow the handler should be added with the argument `-user_rule` and value `CHECK_EXISTS`. To change the functionality the function `callCustomerRuleHandler` from the ECM mapping file has to be modified.

The rule handler `T4X-attach-LogfileDataset` is included in each standard T4x workflow template and it does not need an argument. The purpose of this rule handler is attaching a log file to each workflow.

After the workflow has finished, the attached file can be found as follows:

- Select the Item Revision in "My Teamcenter".

- Right-click the Item Revision and select "Send to".

- Select the application "Workflow Viewer".

- In the "Workflow Viewer" click the icon "Attachments" (in the lower left corner).

- Double-click the Dataset *<T4x>-Logfile*... to view the content of the log file.

2.1.5 Obtain Detailed Information about the Workflow

The `<Obtain Info from WF - t4x_triggering_get_info_form_wf_doc.sd>` shows how to obtain additional information about the current workflow. Note that:

- Some information may exist more than once so it has to be asked with a counter, e.g. the zero in `::TcData(Workflow:ReleaseStatus:0)` for the first one in the list (second one is :1 etc.). Be sure to set the counter according to your requirements; maybe you have to use a loop to determine the latest one. For the example with the Workflow Job release status (which is not the same as a release status of a Workflow target) it should not be a problem because you know from the Workflow design in which order the statuses have been attached to the Workflow Job.

- The `last_mod_user` of the Workflow Job is the last Teamcenter user doing anything during this Workflow job, e.g. a Reviewer or an automatic user for the T4x job functionality.

For workflow transactions T4x provides the following way to get the `owning_user` and the `owning_group` of the executed workflow:

```
if {[info exists ::TcData(WorkflowJob:owning_user)]} {
  set OwningUser [tpco_formatHEX16 $::TcData(WorkflowJob:owning_user)]
}
if {[info exists ::TcData(WorkflowJob:owning_group)]} {
  set OwningGroup [tpco_formatHEX16 $::TcData(WorkflowJob:owning_group)]
}
```

T4x may read "Personal Attributes" of the owing user of the workflow, too, as they are the stored in the organization hierarchy of the Teamcenter database (Administration > Organization):

```
if {[info exists ::TcData(WorkflowJob:owning_user:PersonName)]} {
  set OwningPerson \
    [tpco_formatHEX16 $::TcData(WorkflowJob:owning_user:PersonName)]
}
if {[info exists ::TcData(WorkflowJob:owning_user:PersonAttribute:PA1)]}
{
  set PA1 \
    [tpco_formatHEX16
$::TcData(WorkflowJob:owning_user:PersonAttribute:PA1)]
}
if {[info
exists ::TcData(WorkflowJob:owning_user:PersonAttribute:PA11)]} {
  set PA11 \
    [tpco_formatHEX16
$::TcData(WorkflowJob:owning_user:PersonAttribute:PA11)]
}
```

The `owning_user` from code example above of a workflow job means the Teamcenter user who initiated the workflow job. The `last_mod_user` of the Workflow Job is the last Teamcenter user doing anything during this workflow job, e.g. a Reviewer or an automatic user for the T4x job functionality.

Caution:

- The T4x mapping can check the current workflow task name, but not the handler name!
- If it seems that a T4x workflow handler did nothing in a workflow job because there was no valid target in the current workflow job (e.g. the T4x document handling is configured for standard ItemRevs only, but a 'MyItemRevision' was selected as the workflow target), this may be not obvious at once, because T4x will not write a transaction log for that processing.
  - It may be checked with the workflow log file Dataset attachment (by the rule handler T4X-attach-LogfileDataset):
  - If the target is correct, the workflow log file shows e.g. (12345 is the Item ID):

```
<T4x>-transfer-DocumentInfoRecord: returns for Action:
  Complete => CREATED

<T4x>-transfer-DocumentInfoRecord: process MSWord object:
  12345/A

<T4x>-transfer-DocumentInfoRecord: returns for Action:
  Complete => UPDATED.
```

  - If there is no correct target, then the line with the Item ID is missing, so in case of a problem that is the easiest way to check if T4x processed the object at all.

2.2 External Triggers

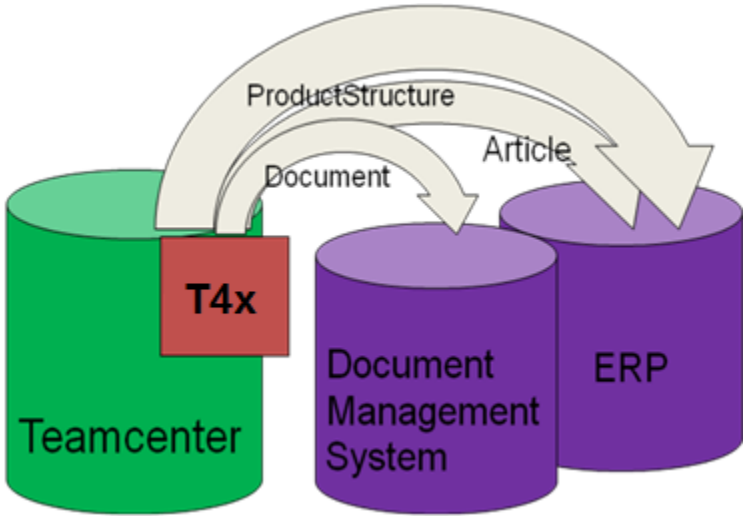
The transfers from Enterprise Application to Teamcenter can be executed synchronously or asynchronously. The asynchronous transaction means that no immediate feedback should be send to the EA-event trigger back. In contrast to asynchronous call, the synchronous will wait for the response and block the execution of the next transactions. Keep in mind that it can take a while until the data transfer is completed.

For more information see chapter *Web Services* in the **Active Integration - Connectivity Guide**.

3. T4x Export Concept

Perhaps the simplest, but also the most common integration is to transfer data resulting from an engineering process in Teamcenter to the EA, e.g. an ERP system. This kind of transport is called export or transfer in the context of T4x. The type of data transferred may vary widely: You can export meta data (items and revisions of different types), structured data (BOMs) and documents (datasets and files) from Teamcenter.

On the EA side, you can decide to update article, part, document and structure data depending on the capabilities and data model of the EA. Depending on the complexity of your scenario, it may be necessary to distinguish several different types of transfers, e.g. because for different item types there are different attribute mappings, typically resulting in different target types in the EA. In T4x you can give each transfer type its own name, the "TargetTypeName". By specifying different "TargetTypeNames" you could for example transfer the PDF attached to a revision to the company's document management system whereas the meta data of the revision creates an article in the ERP system.



In the mapping files you can distinguish the "TargetTypeName" as it is one of the input parameters to all the procedures. The procedures operate on a special data structure (sometimes called "TcData") that contains all the Teamcenter data being relevant for the current TargetTypeName and the selected Teamcenter object - usually a single revision and its attachments. You can access data in this structure by calling T4x procedures in your mapping.

You can influence the amount of data in this structure by setting preferences in Teamcenter. These preferences tell T4x which types are allowed for which TargetTypeNames and which navigation paths from the initial revision should be followed for their evaluation. It is possible to follow Generic Relationship Management (GRM) relations (e.g. to access datasets, master forms, related revisions) and object links (e.g. to access item attributes, or for the unit of measure). You can even define which attributes should be evaluated and which attributes should be ignored, if you need this level of detail.

The T4x configuration for data transfer always consists of these steps:

1. Configure the data in the Teamcenter preferences.

2. Specify the mapping (define the handling of the data, e.g. which Teamcenter attribute gives data to which field EA and vice versa) in the mapping files.

3. Create the compilation of all mapping source files:
<t4x>_mapping_config.rfdt and **t4x_mapping_config.rfdt**.

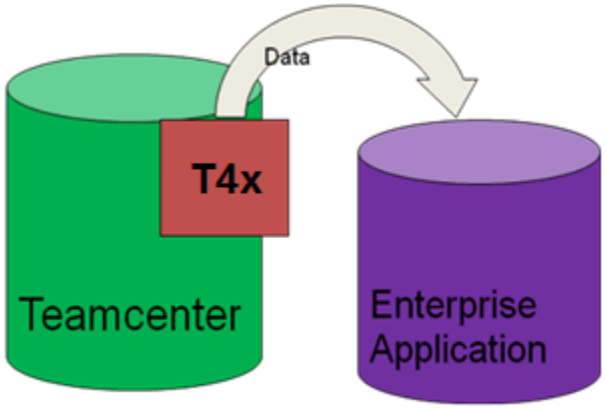
4. Move these file to **<T4x_ROOT>\lib**.

5. Restart T4x GS.

Transfers can be triggered by:

- Workflows: The product specific workflow action handler `<T4x>-transfer-GenericObject` (can also handle BOMs) can be integrated into any Teamcenter workflow. The actual transfer can then take place synchronous or asynchronous to the user's workflow.

- Programmatically: TCL procedures in the namespace `::T4X::BATCHJOB::EXPORT` allow direct, synchronous execution and asynchronous execution in a T4x job. These calls are used in the GS Admin UI in the scripts *<T4x> transfer test* and *<T4x> BOM transfer test*. Here you can find examples of how to use these calls. Using these calls in a procedure exported as web service (SOAP or PXML) you can trigger any transfer by web services. Based on the scripts it is also possible to implement mass data exports, batch exports, command line utilities, etc.



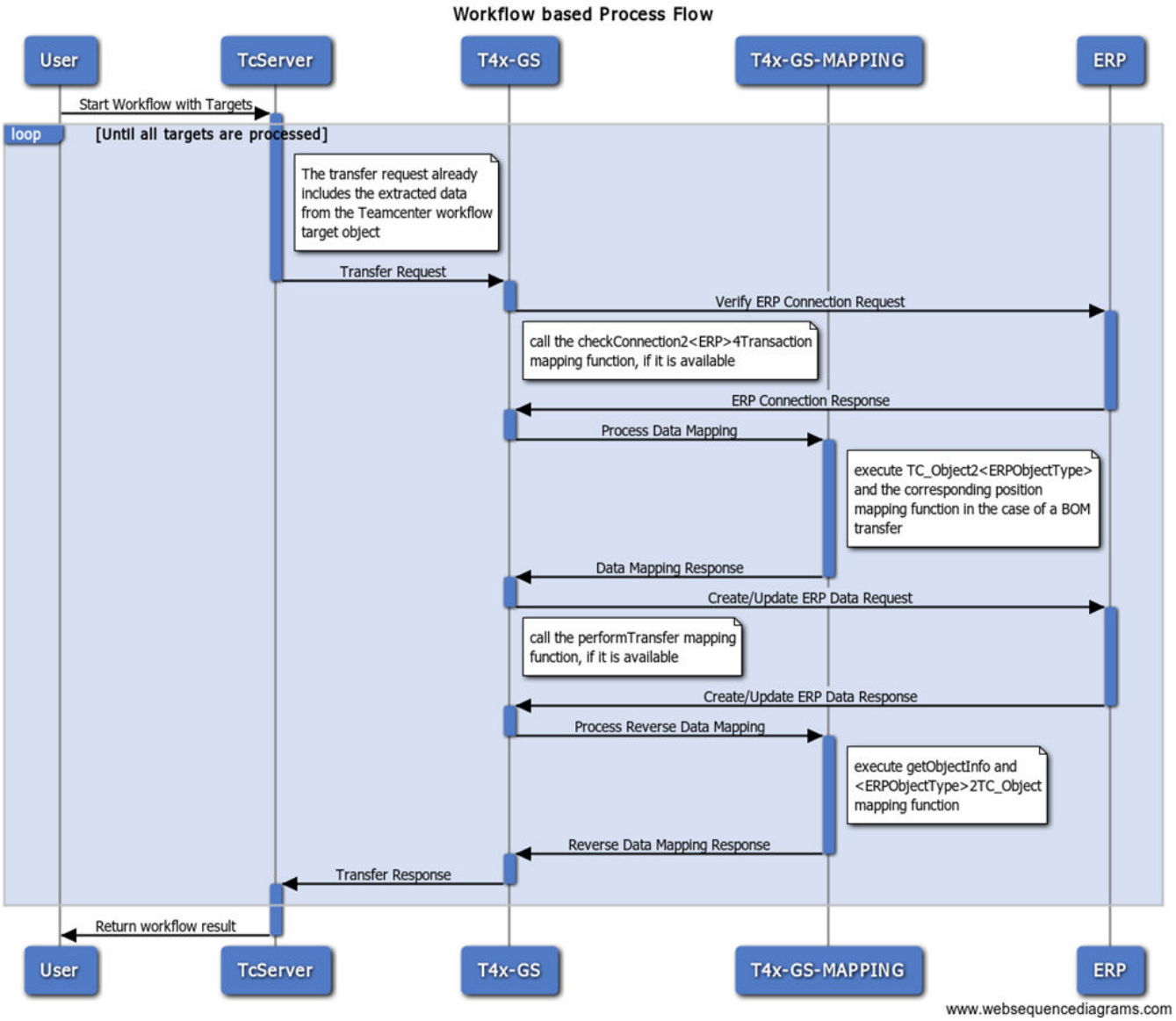
- **Relationship: Preference-Mapping-Handler**.
- **Generic Preference Concept**.
- **Generic Mapping Concept**.
- **Object Transfer**.

- **Structure Transfer**.
- **Error Handling**
- **Advanced Configuration**.

Sequence Flow

As a sample (see figure bellow) shows the execution sequence of a workflow triggered transfer job in a generic way (valid for the T4x product group T4S, T4O and T4EA). There are some parties involved:

- User: The Teamcenter RAC user.
- TC-Server: The Teamcenter server process assigned to the user session.
- GS: The T4x Gateway Service.
- GS mapping: The custom part of the GS, the mapping (separated here to show the custom mapping procedures on a separate line. In reality the mapping is not executed in a separate active instance, but within the GS).
- EA: The EA system.



3.1 Relationship: Preference-Mapping-Handler

To be able to transfer a particular transfer type (object or structure) you have to execute the following steps:

1. **Modify Teamcenter preferences** to describe which Teamcenter data T4x should read and to which Teamcenter objects T4x should write.

2. Adapt the mapping code in TCL dedicated to this type of transfer, mainly to define the **mapping** and or **reverse mapping** between Teamcenter and EA fields.

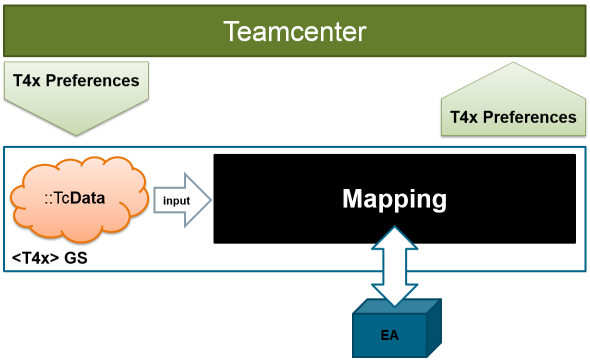
3. Use **Trigger the handler** for a particular transfer type.

3.2 Generic Preference Concept

The preferences are settings stored in the Teamcenter database. They provide a mechanism to control and define Teamcenter behavior by letting you modify the interface, set default behavior, and specify default values. Some preferences are read at start-up or when you log on to Teamcenter, and others are read during Teamcenter application usage. Preferences let you configure many aspects of a session of application behavior. It can be such as how assemblies are revised, whether extension rules are bypassed for specified operations, and which Teamcenter objects are displayed in integrations. In general, the using of the preferences is essential for bi-directional data exchange.

Configure the data in the Teamcenter preferences (set the Teamcenter data objects whose data are read by T4x or may be written to Teamcenter, respectively). The basic configuration of T4x is done by adding certain keys to the Teamcenter preferences (*Teamcenter – My Teamcenter – Edit – Options...*) that are stored in the Teamcenter database itself. Some T4x preferences have been imported while installing T4x using the TEM. Unless stated differently, the every configuration key described in this document is valid for every supported Teamcenter version.

The following graph shows the overview to the generic concept of data exchange between Teamcenter, T4x and EA



- **Relationship: Preference-Mapping-Handler**
- **Preferences**
- **Blacklist/Whitelist**

3.2.1 Preferences

The first step you have to do is configuring the data sources, i.e. the Teamcenter data T4x will read in order to pass the data to EA. This is done in the Teamcenter preferences.

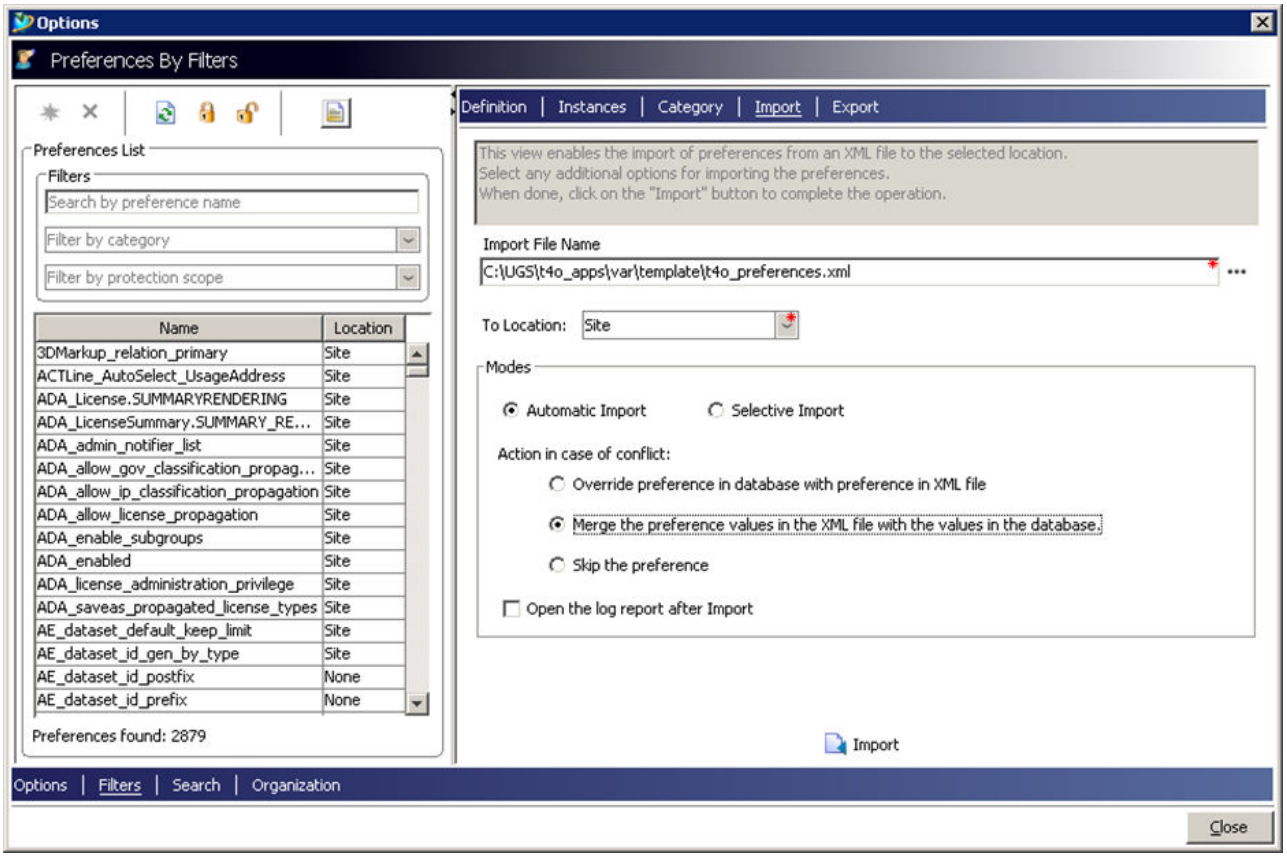
In order to perform large modifications, do not modify each preference directly in Teamcenter but do the following:

- Export your existing Teamcenter preferences.

- Create a backup of the export file
- Modify all the affected preferences in that file according to your needs.
- Delete all the other (unmodified) preferences from that file.
- Import the modified file to Teamcenter as described below.

There are two different ways to set Teamcenter preferences to modify the configuration:

- Using an XML file with the preferences
- Import the saved Teamcenter preferences from a file that may have been exported from another Teamcenter installation. Using RAC, select the menu Edit – Options in the application "My Teamcenter", then:
  - In the lower left part, click "Filters".
  - In the upper-right part of the window, click "Import".
  - Click on the Browse button (the three dots) on the right of the "Import File Name" field and browse to the XML file with your customized T4x preference definition.
  - Select the other options as shown in the screenshot (Site, Automatic and Merge):



- In the lower right part of the window, click "Import" and wait some seconds until it is ready (status in the lower left corner will switch from "Importing..." to "Ready").

- If you have group or role specific T4x preferences (user specific preferences are not supported), do the same with them: be sure to have only those preferences in the import XML file and select the appropriate "Location" during the import. T4x also handles group and role preferences but in order to not allow Teamcenter users giving themselves more rights than defined by an administrator, T4x does not handle user preferences.

- Restart Teamcenter in order to activate the modifications. Then, T4x is ready to work, that means you should find the additional menu "*T4x Gateway*" with some sub menus in the Teamcenter GUI.

- Define the required T4x preferences carefully. Only define and import preferences that are required to support your custom types and use cases. Defining more preferences than needed may confuse you and may result in performance loss.

The most important preferences for the T4x transfer are:

Teamcenter 4 Target System Preferences

- `<T4x>_<TargetTypeName>TypeList`

- `<T4x>_<TargetTypeName>Mapping4<TC_Object> Revision`

Caution:

- If you try to set invalid data to an attribute in a Teamcenter form, Teamcenter may refuse the whole saving action or fail to save the form that contains the doubtful data.
- If you try to fill a Teamcenter attribute with a string that is too long for this attribute, the string will be cut to the correct length and stored. The logs will contain a warning, but the user is not notified by a message box. Most of this data is not stored in clear text format. If you want to output them, you have to use the code sequence `[binary format {H*} $var]`.
- If a Teamcenter attribute is not defined as type "String", you have to set the data in a special format in order to store them in Teamcenter:
  - Logical (Boolean): set "TRUE", "YES" (not case sensitive) or "1" in the mapping file to set the value to „true". Any other string will set the value to „false".
  - Date: use the format "2020.06.30 15:07:00" (for setting it to June 30, 2020 at 7 past 3 PM), might be different depending on the Teamcenter installation.
- From a T4x point of view, there is no problem with a Teamcenter attribute defined as type "LongString"; just use it as any other string attribute. If the string to write to this attribute contains a new line ("\n"), Teamcenter will automatically put the following part of the string into the next line.
- If you want to write information to customer specific attributes on the Item or Item Revision, please make sure that the attribute of the "Business Object" in BMIDE is configured as modifiable.

Target System 2 Teamcenter Preferences

For the "reverse mapping" the target sources in Teamcenter preferences can be configured, i.e. the Teamcenter attributes T4x should be able to write to. The `<T4x>_<TargetTypeName>Mapping2<TC_Object>` preferences are not mandatory for the export. T4x provides a separate mapping function `<T4x>_<TargetTypeName>Mapping2<TC_Object>` for each data type for the reverse mapping of data from T4x to Teamcenter. If you call such a function for a Teamcenter data target that is not configured correctly, it will not be modified.

If there are no reverse mapping data defined for a Teamcenter object that is defined in the `Mapping2` preference, then there will be a warning in the logs as well:

```
Update of Teamcenter objects was skipped. Reason: No values found for
target
object >ItemRevision:items_tag:Item<
```

The attempt to set data to a Teamcenter data object that does not exist is ignored and does not cause a problem. This is useful when setting an attribute that exists in an object of one Item Type but not in another Item Type. So you do not have to distinguish between the Item Types explicitly.

The attempt to set Teamcenter data that have not been configured correctly (missing or insufficient preference `<T4x>_*Mapping2*` or a misspelling in the corresponding preference name or value) will result in the correct mapped data but as it impairs performance T4x will report an error, e.g.

```
Status : 1700 = Option not found:
<T4x>_ArticleFieldMapping2items_tag:Item
```

3.2.1.1 Extract all Available Sequences of Revisions from Teamcenter

Beginning with T4x 11.4 it is possible to extract all available sequences of Revisions by specifying logical preference `T4X_ingnoreInactiveItemRevisionSequences` with the value `true`.

3.2.1.2 Disable the Extraction of Inactive ItemRevisions

The default T4x data extraction behavior extracts all available sequences of Revisions if they are referenced by a given start object (like a Solution Items under a Change object). This results into a more complex mapping implementation to identify the correct object sequences in the mapping. Beginning with T4x 11.4 it is possible to ignore the inactive ItemRevision sequences during the data extraction by specifying logical preference `T4X_ingnoreInactiveItemRevisionSequences`.

Example:

```
T4X_ingnoreInactiveItemRevisionSequences=
true
```

3.2.2 Blacklist/Whitelist

Black- and Whitelists limit the data that is read out of Teamcenter, increase performance and reduce memory usage.

Add the value `#__getAllProperties__#:Properties` to the object's `<T4X>_<ObjectName>Mapping4<TC_object>` preference, to allow access to all attributes, e.g.

```
<T4X>_<ObjectName>Mapping4ItemRevision=
IMAN_master_form_rev:ItemRevision Master
#__getAllProperties__#:Properties
```

Be sure to **always limit** the Teamcenter data T4x should actually read from Teamcenter. There may be a lot of data and some data that require a complex search in the whole Teamcenter database (e.g. where-used data). Every object of that type as described here so that it takes not too much time until T4x received all the data from Teamcenter.

Additionally, Teamcenter internally can define some data that are not allowed to be read. In this case, the T4x transaction may end with the following Teamcenter error message:

```
The application is not privileged...
```

Check in the Teamcenter syslog file which data Teamcenter does not allow to be read, and configure T4x so that it does not try reading that data from Teamcenter. See more examples for those preferences in the following files in the T4x original package (may be deleted by the installation) in *<T4x_ROOT>\var\template\var\t4xBMIDE\full_update\<t4x>_install.zip*:

install\t4x\preferences see the files **_black_and_white_list_preference_defaults.xml*

Set the following preference (single value) to tell T4x only to read limited set from the currently processed Teamcenter object:

```
<T4x>_<ObjectName>PropertyListMode4ItemRevision=2
```

That means whenever T4x handles the Teamcenter object of type `ItemRevision` in the `<ObjectName>` functionality, it has to use the property list mode "2". Other possible examples are:

```
<T4x>_<BillOfMaterial>PropertyListMode4view

<T4x>_<Document>PropertyListMode4<TC_Doc> Revision
```

The possible values for those "property list mode" preferences are:

- 0: process all properties, so it behaves the same as if it is not set.
- 1: use black list preference `*Property2IgnoreList4*` (e.g. `<T4x>_<ObjectName>Property2IgnoreList4ItemRevision` or `T4X_Property2IgnoreList4ItemRevision`) to tell T4x the properties it should not read from the current object.
- 2: use white list preference `*Property2ProcessList4*` (e.g. `<T4x>_<ObjectName>Property2ProcessList4ItemRevision` or `T4X_Property2ProcessList4ItemRevision`) to tell T4x the properties it should read only from the current object.

As in most cases, there are a couple of known properties to read only, use value `2` preferably.

If you do not want to define different behavior for MM, BOM etc., set the following preference instead of the previously defined one that defines the property list mode for a specific Teamcenter object type only but the same for all the target system object types:

```
T4X_PropertyListMode4ItemRevision=2
```

That means whenever T4x handles a Teamcenter object of type `ItemRevision` in any type specific functionality (Material or Product, BOM, Change Management Object, Document ...), it will use the property list mode `2`.

This is very useful for reading the same Teamcenter properties in any kind of transaction for the same Teamcenter object type, e.g. the Teamcenter status.

Then define all the Teamcenter properties you want to read during a transaction, e.g.:

```
<T4x>_<ObjectName>Property2ProcessList4ItemRevision= object_name
item_revision_id ics_classified creation_date last_release_status
```

This defines the properties you want to read, i.e. if the preference *`PropertyListMode4`* value is `2` (use white list). If it is `1` (use black list), you have to set the following preference in order to tell T4x which properties it should not read instead. **Example:**

```
<T4x>_<ObjectName>Property2IgnoreList4ItemRevision= object_desc
ics_classified creation_date AuDProjectDocumentView proj_assign_mod_date
checked_out_user external_apps IMAN_References
```

Same as with `<T4x>_<ObjectName>PropertyListMode4ItemRevision`, you may specify `T4X_Property2ProcessList4ItemRevision` instead of `<T4x>_<ObjectName>Property2ProcessList4ItemRevision` or `T4X_Property2IgnoreList4ItemRevision` instead of `<T4x>_<ObjectName>Property2IgnoreList4ItemRevision`, respectively.

> Caution:
>
> Be sure to not set any Teamcenter properties in the white list that do not exist;
>
> so maybe you have to define the preferences `*Property2ProcessList4*` differently for the different object types that are needed
>
> - If the white-list is configured incorrectly, it may happen that T4x cannot read any properties for the object at all, so check those preferences in detail in case of such a problem.
> - In principle, this refers to the black list (instead of the white list) as well.

3.3 Generic Mapping Concept

- **Mapping (Teamcenter data to EA)**
- **Reverse Mapping**
- **Workflow Arguments in the Mapping**

3.3.1 Mapping (Teamcenter data to EA)

A mapping file contains implementations for the customer specific parts of a T4x transfer or import. It is written in the scripting language TCL and has to contain implementations for certain procedures. A very useful help on TCL can be found on the following Internet page: **http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm**.

Separation of the mapping logic and the technical connectivity should manifest itself in the structure of the configuration source code. It has proven useful to separate the call of the external system from the data mapping and the Teamcenter calls by encapsulating the technical interface in a separate layer (e.g. in a separate TCL function or separate module). You can pass parameters and return values as simple function parameters, associative TCL arrays or TCL dictionaries (dict). Building the (XML) payload of an interface call and parsing the return value should also be separated into functions. General parameters like the web service URL, the database connect string or user credentials of a technical user can be stored in associative TCL arrays in the custom namespace or globally in *<t4x>_mapping_config.sd* to be available wherever you need them. Alternatively you may use the T4x connection handling to manage all the connection parameters and supply user login dialogs, etc.

Generally you just add, if necessary, your attribute mappings and more complex mapping logic, if necessary. T4x takes care of calling these procedures in the correct sequence and at the right time. It passing the correct parameters, extracting data from Teamcenter and writing your results to Teamcenter. Please see the API Reference for more details on the syntax and semantics of the required procedures.

It is important to understand that such a mapping is a logical unit and each transfer type is given a logical name. This allows the transfer to be started in different ways and from different triggers. Usually you will never call the procedures in a mapping file directly. Instead of that you use a generic TCL call to start the transfer and pass some parameters. T4x will then take care to call the mapping procedures from your mapping files correctly.

A set of templates and sample mapping files are contained in the software delivery of T4x to allow easy start-up.

The function `::T4X::TC::MAPPING::FieldMapping` allows getting Teamcenter data and uses to get the object. For the structuted data use the function `::T4X::TC::MAPPING::IndexedFieldMapping` instead. If you call such a function with a Teamcenter data source that is not configured correctly (maybe a misspelling), the result will always be an empty string.

- The result is always a string, no matter of the data type in Teamcenter.
- If the original data is of type Boolean, you will get "1" (true) or "0" (false).
  The result of a "Date" or other non-string-attribute may be different from what you see in Teamcenter itself.

- It is no problem if you try and read data that does not exist. In this case, the result will be empty. This is useful when reading an attribute that exists in an object of one Item Type (then you get the correct data), but not in another Item Type (then it is empty). So you do not have to distinguish the Item Types explicitly.

- The mapping result is empty if the queried data do not exist (no error), but there is a warning in the T4x session log. If you try to read the not existing attribute on the `ItemRev Master Form`, you will get the message like this:

```
::T4X::TC::MAPPING::FieldMapping - Warning: No TcData entry found
for >Item Revision:IMAN_master_form: Item Revision Master<
and >does_not_exist<!
```

Be aware of the return value of the mapping functions. The return value is T4x specific, and not a part of generic TCL. T4x evaluates the return codes of its functions. If you want to exit a function without causing T4x starting its error handling, be sure to use the same return value that you find at the end of this function and nothing else. In the functions for the mapping from Teamcenter to the EA it can be `OK` or `$Status`.

If you want to stop the transaction and output your own error message, use the return value `ERROR` instead. Then the transaction is aborted completely. If it was started from a workflow, it will stop at this point. To create an own error message, T4x provides two functions:

```
::T4X::CORE::storeMessage2 Mapping $err_msg ERROR
::T4X::CORE::storeUserMessage ERROR <err#> <p1> <p2>
```

Add one of them immediately before the return "ERROR", where the variable `$err_msg` contains your desired error message (set it before). For more details see chapter **Write Custom Log Messages**.

- Basically, function `storeMessage2` is meant for writing error messages into the T4x session log file and the T4x transaction log file as well. For interactive transactions, these error messages will also be displayed in the GUI. If you are using the rule handler `T4X-attach-LogfileDataset`, this function also writes to the workflow log file.

For writing info messages (regardless of an error) into the workflow log file (created by rule handler `T4X-attach-LogfileDataset`), use function `storeMessage2DatasetLogfile`, e.g.:

`::T4X::CORE::storeMessage2DatasetLogfile "MAPPING" "Test message"`. For more details see chapter **Write to the T4x Workflow logfile Dataset**.

With this function the messages are also written into the T4x session log file.

- The parameters for function `storeMessage2DatasetLogfile` are the same as for `storeMessage2`.

- If the parameter `Type` of function `::T4X::CORE::storeMessage2` is set to `ERROR`, the message is written into the *apps_error.log* file additionally (shown in red).

- In the T4x session log file, all messages of type `ERROR` are highlighted in red.

- If you want to skip one object without influencing the rest of the transaction (and not stop the workflow), use return `SKIPPED`. This will not call the `performTransfer` procedure and so no transfer to the EA takes place with the current object data, structure or only one position in the structure. It will not perform the reverse mapping for it, but immediately go on with the next object without providing an error message (which would cause the workflow to stop). We recommend providing a log message in such a case for being able to track why the object was skipped.

- If you want to update your Teamcenter object with the corresponding data from the EA, but not modify the EA data, use return `REVERSEMAPPINGONLY`.

- The functions for the reverse mapping (from the EA to Teamcenter) should always return `OK` as long as no error occurred; else the reverse mapping is skipped and a corresponding log message is created. The update of the Teamcenter object after the transfer was switched off by the reverse mapping status. Only the reverse mapping with return value `SKIPPED` will end the transaction successfully and do no reverse mapping to Teamcenter. With any value besides OK and `SKIPPED`, T4x will treat the whole transaction as if an error occurred actually.

- The TCL procedure `callCustomerRuleHandler` which is included in every mapping file to evaluate specific rule handlers, always needs the return value of the `$Decision`. This variable can be set with the value of either `EPM_go` (workflow proceeds) or `EPM_nogo` (workflow stops).

Be sure to have the Teamcenter attributes configured in the Teamcenter preferences, otherwise T4x will not store any data there. The input parameters of the reverse mapping function are the T4x function name (TargetTypeName, e.g. Article, MM or Item), the status and the corresponding external object number. With this object number an enquiry can made to the EA.

3.3.2 How to Read and Write a Teamcenter Preference in the Mapping

Read a Preference

There are two ways to read the Teamcenter preferences. One way is to read the preference dynamically with an ITK function. Another possibility is to adapt the T4x preferences so that defined Teamcenter preferences are exported for the ERP transfer (recommended).

Reading a Preference for the Teamcenter to ERP Transfer

The content of a preference will be exported along with the object data.

Preference name:

`<T4x>_<TargetType>Mapping4Preferences`

Preference value syntax:

`<PreferenceScope>:<PreferenceType>:<PreferenceName>`

Supported preference scopes:

- all
- user
- role
- group
- site
- system

Supported preference types:

- string
- logical
- integer
- double

The example shows Teamcenter preference settings for the T4S MaterialMaster transfer:

```
T4S_MaterialMasterMapping4Preferences=
group:string:T4X_CustomPref_GroupDefaultPlant
group:string:T4X_CustomPref_GroupDefaultPlant
site:string:T4X_CustomPref_Plants

T4X_CustomPref_GroupDefaultPlant=
1000

T4X_CustomPref_Plants=
1000
1100
```

The content of the selected preferences will be part of the `::TcData` array of the transaction. To access the data in your mapping, use the function `::T4X::TC::MAPPING::getPreferenceValue`:

```
set PlantList    [lindex [::T4X::TC::MAPPING::getPreferenceValue \
```

```
  site string T4X_CustomPref_Plants] 1]

set DefaultPlant [lindex [lindex [::T4X::TC::MAPPING::getPreferenceValue
\
  group string T4X_CustomPref_GroupDefaultPlant] 1] 0]
```

Caution:

For PLMXML based transfers this functionality is not supported. As a workaround it is possible to get the preference values via handler argument like – `<ArgName>=usePreference:<PreferenceName>` (supported are string preferences only).

E.g. add argument `-t4xPlants=usePreference:T4X_CustomPref_Plants` to the workflow handler `T4S-transfer-MaterialMaster`.

To access the data in your mapping, use the function `::T4X::TC::MAPPING::getWorkflowArgumentValue3`:

```
set PlantList [lindex
[::T4X::TC::MAPPING::getWorkflowArgumentValue3 \
  "-t4xPlants"] 5]
```

Reading the preference dynamically with an ITK function:

T4x allows reading the Teamcenter preferences and evaluate them in the mapping with the ITK call `::ITK::getPreference`.

Caution:

This function can only read site preferences.

The function `::ITK::PREF_ask_char_values_at_location` can read a preference by scope. Supported scopes are:

- user
- role
- group
- site

Caution:

If the preference does not exist for the given scope (e.g. user), Teamcenter may answer the value of the preference with the same name but higher scope (e.g. site).

Write to a Preference with ITK

T4x allows to write to a preference via ITK function `::ITK::PREF_set_char_values_at_location`.

Supported scopes are:

- user
- role
- group
- site

3.3.3 Reverse Mapping

To implement the reverse mapping you need to configure the objects and attributes where T4x shall store information after the object transaction has been performed.

To define data mapping from the EA to Teamcenter the following adaptations in the Teamcenter preferences have to be done:

- Definition of Teamcenter objects, which are used to store EA data generally (the definition has to be done for each item revision type supported by T4x).
- Definition of fields used in these Teamcenter objects to store the data (you can define fields for each structure defined in step one).

The data mapping of the EA back to Teamcenter is carried out at the end of every transaction that was not aborted with an invalid transfer status like `SKIPPED` or with an invalid mapping status.

The function `storeReverseMappingAttribute` stores data into a Teamcenter attribute (or property, respectively).

The following example will store the `$ERPMatNo` into the Teamcenter attribute `user_data_2` of the `Item Revision Master Form`:

```
::T4X::TC::MAPPING::storeReverseMappingAttribute
MaterialMaster $ItemRevMaster user_data_2 $ERPMatNo
```

Another important data is a current time stamp. This can be created by TCL using `clock`, e.g.

```
::T4X::TC::MAPPING::storeReverseMappingAttribute
MaterialMaster $ItemRevMaster item_comment
"$TargetTypeName created in ERP -
[clock format [clock seconds] -format "%Y-%m-%d %R"]"
```

If EA data has to be stored to Teamcenter, the function `storeReverseMappingAttribute` has to be used with an EA data source.

T4x shows the data it tries to write to Teamcenter as well as the result of those writing attempts in the T4x transaction log file. For better overview, the time stamp prefixes have been skipped here:

```
UPDATE:SAP2_T4S_Item Revision:IMAN_master_form:SAP2_T4S_Item
Revision Master:SAPMatNo => 2474
Communication to T4X Gateway was successful
Start update of Teamcenter objects ...
Update of SAP2_T4S_Item Revision:IMAN_master_form:SAP2_T4S_Item
Revision Master->SAPMatNo to "2474" was successful
```

The blue "Update" lines show the data as they are set in the reverse mapping function, no matter if the corresponding objects are defined in the `Mapping2` preferences. The green "Update" lines show the data that T4x actually tries to write to Teamcenter. So if an object is not defined in the `Mapping2` preferences but you set a reverse mapping line to write on it, T4x will show it in such a blue "Update" line but not try to actually write it and therefore not show it in a corresponding green "Update" line.

If T4x should write into a Teamcenter an object which is not related to the current object, then its object tag must be used for that functionality; i.e. check it in Teamcenter and use it in the corresponding T4x mapping file.

- Set a `*mapping2*` preference value to a relation/object combination that does not really exist with the used Teamcenter objects (else the actually existing object would be modified instead of the other one), but it must be allowed in the used Teamcenter data model. Example for T4S BOM handling:

  ```
  T4S_BillOfMaterialMapping2view=EPM_referenceAttachment:E4_FormType
  T4S_BillOfMaterialFieldMapping2EPM_referenceAttachment:E4_FormType=
  E4_BOMArray
  ```

- Then use that defined path in the T4S BOM mapping and append the object tag. Example:

  ```
  ::T4X::TC::MAPPING::storeReverseMappingAttribute BomHeader \
    "EPM_referenceAttachment:E4_FormType:$<FormTag>" E4_BOMArray $value
  ```

- Then T4x will write the value `$value` into the property `E4_BOMArray` of the object which has the stated object tag.

3.3.4 Workflow Arguments in the Mapping

The T4x allows evaluating a workflow argument in the mapping using the function `::T4X::TC::MAPPING::getWorkflowArgumentValue2`. This allows different behavior in different workflows without modifying the mapping. It does however not allow affecting the workflow by setting `EPM_go` or `EPM_nogo` (this may be done with the function `callCustomerRuleHandler`, see below).

Syntax:

```
set ArgValue [dict get [::T4X::TC::MAPPING::getWorkflowArgumentValue2 \
  "-ArgumentName" UPPERCASE] ArgumentValue]
```

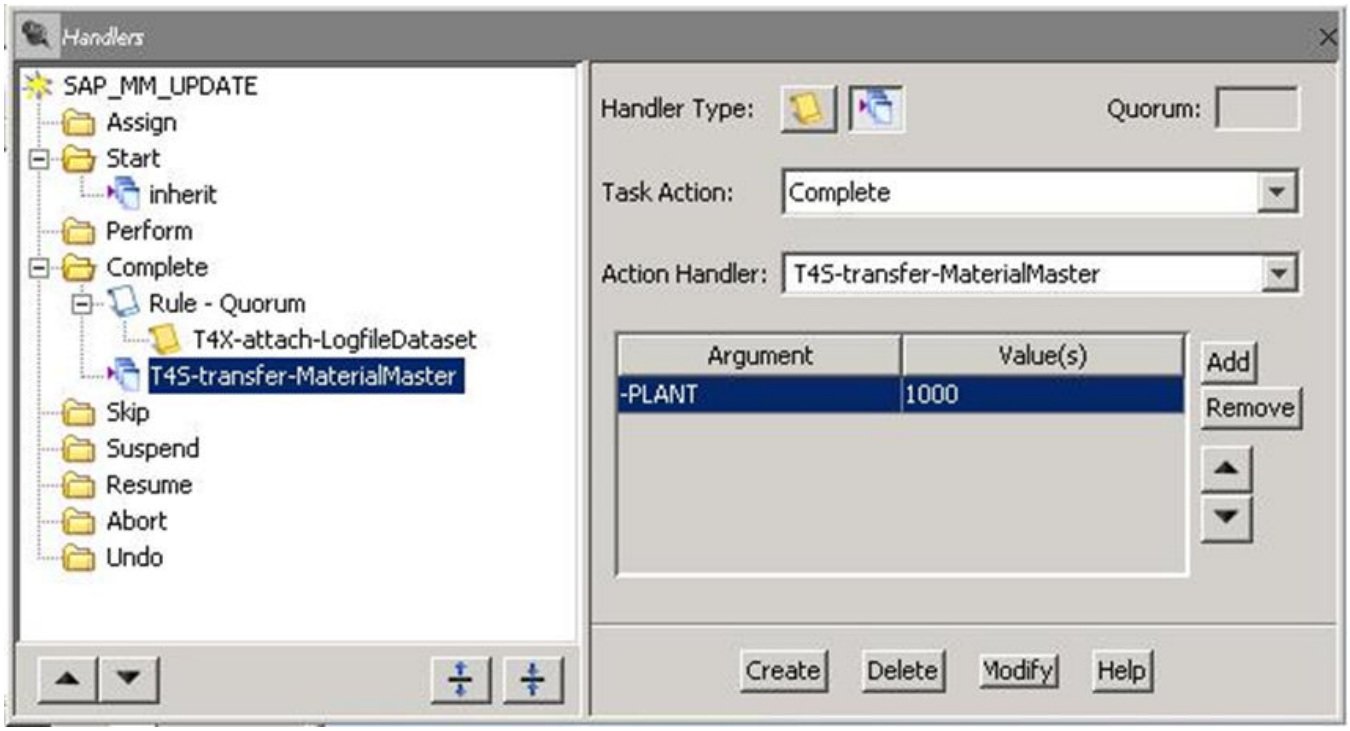
Take into consideration that:

- If the optional Parameter `UPPERCASE` is given as shown here, the workflow handler argument name will be found no matter of its upper and lower case writing in the workflow template and in the function call as well. The value is then stored in whole upper case letters, too, no matter how it is stored in the workflow template.
  Example:
  - The workflow handler argument name in the workflow template is `-MyArg1`.
  - Its value is `TestArgValue1`.
  - When asking the argument value like that:
    `...getWorkflowArgumentValue2 "-myarg1"` then T4x will answer that it could not find the queried argument.
  - When calling the function with the additional parameter `UPPERCASE`:
    `...getWorkflowArgumentValue2 "-myarg1" UPPERCASE` then T4x will answer that it could find the queried argument and store its value as `TESTARGVALUE1`.

Caution:

- For facilitating the handling we recommend only using upper case values (but some of the names are used internally and have to be used exactly as stated in the documentation).
- It is possible but dangerous to search an argument only by its number (`::TcData(Workflow:Argument:x)`), so better use the following method.

Example 1:



Please switch to the <Workflow Arguments Example 1 – t4x_triggering_workflow_arguments_doc.sd > for details.

1. The command `::T4X::TC::MAPPING::getWorkflowArgumentValue2` returns a dict structure with three keys: `Status, ArgumentName, ArgumentValue`.
   The value of Status key will be either `OK` or `NOT_FOUND` (depending whether the requested workflow argument was found or not).

2. Using the T4x "dict" structure (dictionary), T4x allows you distinguishing whether the argument does not exist or it does not have a value by evaluating the `Status` field as in the example above:
   `if {[dict get $MyArg1 Status] eq "OK"} {...` means argument exists.

3. Then the mere evaluation is done:
   `set MyArgValue [dict get $MyArg1 ArgumentValue].`

How to read all workflow job arguments regardless of their names learn in <Workflow Arguments Example 2 - t4x_triggering_get_info_form_wf_doc.sd>.

In this example all arguments are `read`. As they are numbered internally, this can be done with a while loop and an ongoing counter. For each found argument, it does:

- Store the whole argument (which is <arg_name>=<value>) in the variable `T4x_JobArg`.
- Divide the string into the two parts `JobArgName` and `JobArgValue` (the "=" is the separator); the variable `Arg_count` stores the number of parts:

  - `Arg_count` will be 2 in the standard case,
  - `Arg_count` will be 1 if the argument exists but does not have a value (then `JobArgValue` will be empty).
- Output the number of the argument, its name and value.
- The argument counter `ArgNo` is increased
- After the last argument is evaluated, the flag `Args_OK` is set to `false`, so the `while` loop ends.

> Caution:
>
> As one handler is completely processed with all its targets and only then the next handler is executed, the output from different handlers may be far away from the others in the logs. Maybe you will think that not all of them are evaluated correctly.

3.3.4.1 Read Workflow Job Attributes in the Mapping

To read the the workflow's attributes in mapping in four steps:

- Add parameter `-AddObject4Mapping=root_task:EPMTask` to the handler that should read the attributes.
- Check preference `T4X_Property2IgnoreList4EPMTask`: remove the required attribute, e.g. `gov_classification`.
- Check preference `T4X_Property2ProcessList4EPMTask`: add the required attribute, e.g. `gov_classification`.
- In TCL mapping read the job attribute from the TcData buffer: e.g. `set GovClassification $TcData(RootTask:root_task:EPMTask:gov_classification)`.

For detailed description please see **T4x Workflow Handler**.

3.4 Object Transfer

An object is a unique instance of the particular class that has to be transferred from Teamcenter to the EA. It can be a generic Teamcenter item or custom item. Please note that only ItemRevision can be mapped to the target object on the other side.

This step defines which data needs to be read from the Teamcenter source object. It is important to limit the amount of data down to the required data in order to get best performance for your transaction. Any data that is not required for the transaction should not be read! To define the data mapping from Teamcenter to EA the following adaptations in the Teamcenter preferences have to be done:

- What are the required attributes needed to set for each EA object?
- Which information needs to be read from the Teamcenter object to be able to map all attributes to the EA object?
- Is there any information has to be stored back to the Teamcenter object, e.g. status, date of transfer?

We advise to get at least some high level answers to those questions before starting the configuration work. Learn more about the object transfer in the chapter:

- **Object Transfer Specific Preferences**,
- **Read Data of Object**,
- **External Number Assignment**
- **Read Engineering Change Object with different object types and related target objects**
- **Obtain Dataset information**.
- **Read Teamcenter Classification Attributes**

3.4.1 Object Transfer Specific Preferences

The following examples show the complete configuration for the EA object data mapping.

Define the Item Revision types, for example ItemRevision for usage with T4x:

```
<T4x>_<TargetTypeName>TypeList=ItemRevision
```

> Caution:
>
> Be aware some custom ItemRevision types require a blank, standard types do not. Be careful to use the correct syntax.

Specify the TC data objects from which you want to pass data to the EA by T4x:

```
<T4x>_<TargetTypeName>Mapping4<TC_ObjectType>=
items_tag:Item
items_tag:Item:#__getAllProperties__#:Properties
items_tag:Item:IMAN_master_form:Item Master
#__getAllProperties__#:Properties
IMAN_master_form:ItemRevision Master
IMAN_classification:icm0
```

The logic is that T4x follows the relation to the next object and from there continues to the one after that, etc.

Example:

`items_tag:Item:#__getAllProperties__#:Properties` From the "ItemRevision" (which is the current object, stated in the preference name), the "Item" can be accessed following the relation `items_tag`. From the "Item", follow the relation `#__getAllProperties__#` to the properties of the Item.

Note that `#__getAllProperties__#` is not a Teamcenter relation. It is inserted by EA to be able to follow the same logic that a relation leads to the next object. A similar thing is `IMAN_classification:icm0`. The key word `Item Revision` does not have to be stated explicitly as a value, because it is the current object itself. If you do not state `#__getAllProperties__#:Properties`, T4x will only be able to read its properties `object_name`, `object_desc` and `item_revision_id` (or `item_id` for the `Item` itself, respectively)

If you are in doubt whether you configured the data sources correctly, try the T4x script "<T4x> mapping test" on GS.

To improve performance you can use the White or Black List feature, e.g. by setting the preference `<T4x>_ArticlePropertyListMode4ItemRevision=2`.

Next step is the definition of mapping the data to the required EA fields. Please look into the product specific configuration guides for details. For defining the attribute mapping for an object transaction please review the delivered mapping template file. To learn about how to set EA field attributes please have a look into the mapping template files.

In a final step objects and attributes have to be configured where T4x shall store information after the object transaction has been performed.

To define data mapping from EA to Teamcenter the following adaptations in the Teamcenter preferences have to be done:

- Definition of Teamcenter objects, which are used to store EA data generally (the definition has to be done for each ItemRevision type supported by T4x).

- Definition of properties and attributes used in those Teamcenter objects to store the data (you have to define fields for each structure defined in step one).

The data mapping of the EA item back to Teamcenter is done at the end of every successful EA transaction that was not aborted with an invalid transfer status like `SKIPPED` or with an invalid mapping status.

The example describes reverse mapping of data to the ItemRevision Master using the Teamcenter relation IMAN_master_form (some fields of the form) and to the ItemRevision itself (description), as well

as the Item ID. The example contains the handling for different ItemRevision Types, too (ItemRevision and DesignRevision).

Specify the Teamcenter data objects which are allowed to receive data from EA by T4x:

```
<T4x>_<TargetObjectName>Mapping2ItemRevision=
IMAN_master_form:ItemRevision Master
items_tag:Item
items_tag:Item:IMAN_master_form:Item Master
ItemRevision
```

Specify the Teamcenter attributes which are allowed to receive data from EA by T4x:

```
<T4x>_<TargetObjectName>FieldMapping2IMAN_master_form:ItemRevision
Master=
item_comment
user_data_1
user_data_2
user_data_3

<T4x>_<TargetObjectName>FieldMapping2ItemRevision=
object_desc
<T4x>_<TargetObjectName>FieldMapping2items_tag:Item=
item_id

<T4x>_<TargetObjectName>Mapping2items_tag:Item:IMAN_master_form:Item
Master=
user_data_1
```

3.4.2 Read Data of Object

The function `::T4X::TC::MAPPING::FieldMapping` is used to get data for the object. Please take a look at the syntax in the product template file for the particular object. The following example shows how to read the `Item` or `ItemRevision` and maps the values to the keys like `ID`, `RevisionID` etc.:

```
 dict set InputDat ID \
   [::T4X::TC::MAPPING::FieldMapping $Item "item_id"]
dict set InputDat RevisionID \
   [::T4X::TC::MAPPING::FieldMapping $ItemRev "item_revision_id"]
dict set InputDat Name \
   [::T4X::TC::MAPPING::FieldMapping $ItemRev "object_name"]
dict set InputDat Description \
   [::T4X::TC::MAPPING::FieldMapping $ItemRev "object_desc"]
dict set InputDat UnitOfMeasure \
   [::T4X::TC::MAPPING::FieldMapping $Item "ItemUnitOfMeasure"]
```

Get the current Teamcenter object type

At nearly any point in the T4x mapping you can retrieve the type of the current object (which might be for example `ItemRevision` in a T4x object transfer or `MSWord` for Document) with the line:

```
set Object_Type $::TcData(ItemInfo:TypeName)
```

Define Specific Revisions from the current Item to use

By default T4x always uses only the last ItemRevision of the selected Item. In a very special case it could be helpful to select an ItemRevision defined by a revision rule (similar to a BOM revision rule) which is the T4x standard but selecting.

Example:

Set the preference with the value

```
T4S_MaterialMasterMapping4TP8_MM_ItemRevision=

TP8_Rel2Localization:TP8_Plant_Item:use_revision_rule=Latest
Working:TP8_Plant_ItemRevision :T4S_Properties:Properties
```

Read properties from a Teamcenter relation

Since the Teamcenter version 10 it is possible to define properties directly on the relation objects. That way data can be stored only in a specific context. T4x supports reading those attributes.

You need to extend the use case specific preference to read the relation attributes. Then you can access the property from the mapping.

Scenario: On material master transfer, we determine the relevant SAP target plant from Teamcenter plant object, e.g. `SAP2_MEPlant` that is related to the material part with the relation `SAP2_PlantUsage`. The SAP plant ID is equal to the Plant object's Item ID, the plant specific material status is derived from the relation property `sap2_plant_status`. In that way you can store different plant specific status values within Teamcenter and transfer them separately to each SAP target plant.

Preference:

```
T4S_MaterialMasterMapping4SAP2_T4S_Item Revision =

SAP2_PlantUsage:SAP2_MEPlant:#__getAllProperties__#:Properties
```

Mapping:

```
set Plant [::T4X::TC::MAPPING::FieldMapping "$ItemRevisionType: \
 SAP2_PlantUsage:SAP2_MEPlant" item_id]
set PlantStatus [::T4X::TC::MAPPING::FieldMapping "$ItemRevisionType: \
 SAP2_PlantUsage:SAP2_MEPlant" "RelationProperty:sap2_plant_status"]
```

Whenever you define the key `#__getAllProperties__#:Properties` within the preferences to get *all* properties for the given object, it is strongly recommended to use either the Blacklist or Whitelist feature to limit the amount of data that is processed. See **Blacklist/Whitelist**.

Check if a specific Teamcenter attribute exists

The following function allows asking if an attribute in a given form exists, no matter if it has a value. Note that using the regular way of getting an attribute value in the mapping does not distinguish if the queried attribute is empty or does not exist, because the value will be empty in both cases. There will be a warning in the logs but the result is empty in both cases. The following procedures are part of the standard software delivery:

```
::T4X::TC::MAPPING::testFieldExists
::T4X::TC::MAPPING::testIndexedFieldExists
::T4X::TC::MAPPING::testRootTaskFieldExists
```

The meaning is:

- `testFieldExists`

  is the default check for an attribute, see below

- `testIndexedFieldExists`

  checks an attribute in an environment with several target objects, especially in BOM, see the mapping examples there

- `testRootTaskFieldExists`

  checks if data of the Workflow Root Task exist

So you may call them from every point in the mapping, e.g.:

```
if {[::T4X::TC::MAPPING::testFieldExists \
 "$ItemRevisionType:IMAN_master_form_rev:$ItemRevisionType Master" ABC]\
    eq "OK"} {
 tpwrite "The attribute ABC exists"
}
```

Read more than one Teamcenter form of the same type attached with the same relation

In most cases, the combination of relations and types should be unique, e.g. an Item has exactly one Item Master form by default. If you have several targets with the same combination of relations and types and use the T4x default way of reading attributes, Teamcenter will give you data from any one of those targets. In order to set up T4x to read several objects with the same combination of relation and type, set the following in the mapping.

```
<Read Teamcenter form-t4x_export_advanced_config_read_form_doc.sd>
```

Please note that:

- As with every T4x functionality that reads Teamcenter data, you have to specify the searched Objects in the mapping4 preference(s). Remember the easiest way to check their correctness is the test mapping.

- Although the name looks like that, the `parent_object_tag` in the code example above is not the tag of the parent object but the tag of the object that is stated in the logic before the target object. If you retrieve the object tag of an Item with

  ```
  [::T4X::TC::MAPPING::FieldMapping \
   "$ItemRevisionType:items_tag:$ItemType" object_tag]
  ```

  which works as expected, but if you replace the parameter `object_tag` by `parent_object_tag` then you get the tag of the Item Revision. That means that asking for the current object's parent_object_tag will always fail.

- For handling BOM positions, the function name is `findIndexedObjectList4Attribute` instead of `findObjectList`. Please see object and structure mapping description and templates for details on how to handle the differences between "mapping functions" and "indexed mapping functions".

- Unfortunately it is impossible to write into all of these forms in one step up to now.

Determine if a Teamcenter attribute has still its default value (is null)

With the function `::ITK::AOM_is_null_empty` it can be determined, whether a particular attribute is null or not. This is particularly important for non-string attributes, since getting the value of a null attribute returns a default value (like 0) that is indistinguishable from a legitimate user-entered value.

Via the new preference `T4X_enableAttributeIsNullCheck` of type `Logical` and `Single` value (true/fase) it is possible to enable an additional attribute check during the data extraction of Teamcenter data to provide the information as input for the mapping. To check if the value is null, the field name parameter of the `FieldMapping` function needs to be extended with `:IsNull`.

Example:

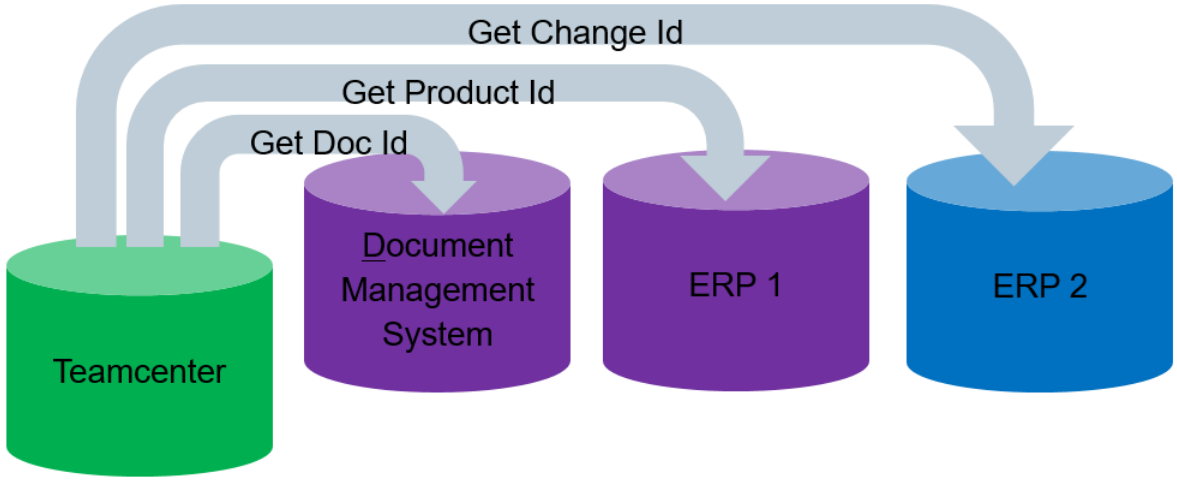
```
TcData(ItemRevision:IMAN_master_form_rev:ItemRevision
Master:user_data_1)=
TcData(ItemRevision:IMAN_master_form_rev:ItemRevision\
Master:user_data_1:IsNull)=true

# Get the value
set PropertyValue [::T4X::TC::MAPPING::FieldMapping $ItemRevisionMaster\
 "user_data_1"]

# Check if the value is null (will be empty or „true")
set PropertyIsNull [::T4X::TC::MAPPING::FieldMapping $ItemRevisionMaster
\
  "user_data_1:IsNull"]
```

3.4.3 External Number Assignment

T4x also provides the support to trigger the external number assignment or number validation for Teamcenter Items by a corresponding extension that is enabled by an extra libt4x_idgen library a set of preferences and a corresponding product specific IDGEN mapping file.



In the mapping files you can distinguish the external number generator that should be called by the Teamcenter Item type name. Beside the Item type information it is also possible to access a limited set of Teamcenter session information (e.g. Username, ...) provided via the TcData buffer as in all other transfers.

The T4x configuration for data transfer always consists of these steps:

1. Configure the data in the Teamcenter preferences:

- `T4X_IDGEN_TypeList=`
  `ItemRevision`

- `T4X_IDGEN_TargetSystem4<TcItemTypeName>=`
  `<T4x>`

- `T4X_IDGEN_Function4<TcItemTypeName>=`
  `T4X_IDGEN_USE_EXTERNAL_NUMBER`

- `T4X_IDGEN_max_loop=`
  `10`

2. Specify the mapping (define the external number generator call) in the corresponding *<t4x>_idgen_mapping_template.sd* mapping file.

3. Create the compilation of all mapping source files (*<t4x>_mapping_config.rfdt* and *t4x_mapping_config.rfdt*).

4. Move these file to *<T4x_ROOT>\lib*.

5. Restart <T4x> GS.

The external Number Generator can be triggered by:

- Teamcenter Number Assign button.

- During the create new Item dialoge.

3.4.4 Read Engineering Change Object with different object types and related target objects

This is useful if you want to use Teamcenter change management and want to copy the correlation you have in Teamcenter between the Engineering Change Management (ECM) and its target objects to EA. Set the following Teamcenter preferences:

Example for the Teamcenter change management:

```
<T4x>_process_CM_pseudofolder_names4ProblemReportRevision=
CMHasSolutionItem
CMHasProblemItem

<T4x>_process_CM_pseudofolder_names4ChangeNoticeRevision=
CMHasSolutionItem
CMHasProblemItem
```

```
<T4x>_process_CM_pseudofolder_names4ChangeRequestRevision=
CMHasSolutionItem
CMHasProblemItem

<T4x>_process_CM_pseudofolder_names4<T4x>_Item Revision=
CMHasSolutionItem
CMHasProblemItem
```

This tells T4x to use the objects as workflow targets that are stored in the corresponding pseudo folders `affected_items` and `solution_items`. Nevertheless, the Teamcenter ECM is the object where the workflow has to be started from, no matter if it is the Item workflow or another one.

Please note that:

- As this is similar to the Teamcenter preferences standard key "CM_pseudofolder_names", you can copy the folder names from there.
- T4x uses a separate preference for each Teamcenter change type (see suffixes in above example: `4ProblemReportRevision, 4ChangeNoticeRevision, 4ChangeRequestRevision and 4EBS2T4O_Item Revision`).

Caution:

- If you add an Item Revision as workflow target linked into one of the ECM pseudo folders stated in those preferences,T4x will process that Item Revision twice. One as target and one more from the relation in the pseudo folder
- In order to avoid that double processing, there are two ways:
  - do not use an Item Revision as workflow target if it is present in such a pseudo folder in the same workflow job already
  - or exclude the pseudo folder name from the preference `CM_pseudofolder_names`
- If the corresponding preference `CM_pseudofolder_names` does not exist or does not have a value, T4x will not use any object from the ECM pseudo folders.

You can use data related like this in any of the T4x mappings, e.g. use the Teamcenter Change Item ID as EA Engineering Change Notice ECN, so you have to read this ID during a BOM transaction although your current object may be one of the Item Revisions linked into one of those pseudo folders:

- Add the following value to the corresponding Mapping4 preference (remember to add it to the Item related preference for BOM, e.g. `<T4x>_ItemMapping4ItemRevision` for the handling in the Item transactions):

```
EC_affected_item_rel:EngChange Revision:items_tag:EngChange
```

- Then add in the reverse mapping:

```
set change "$ItemRevisionType:EC_affected_item_rel: \
 EngChange Revision:items_tag:EngChange"
set ECN [::T4X::TC::MAPPING::IndexedFieldMapping 0 $change item_id]
```

Similar to this example, you may use the other relations as well. Please have a look into your existing Teamcenter change preferences to find the correct spellings.

3.4.5 Obtain Dataset information

This example shows how to obtain information for a Dataset of type MSWord, if the `ItemRevision` is the corresponding object.

Set the following preference at least:

```
T4S_DocumentMapping4ItemRevision=
IMAN_specification:MSWord
IMAN_specification:MSWord:ref_list/word:ImanFile
```

Note that the entry `IMAN_specification:MSWord` enables T4x to read data from the Dataset whereas the second is for the file(s) in the Dataset (attached to the Dataset as named reference `word`). Then you can access the `Name` and the `Description` of the Dataset. Note that a Teamcenter Dataset does not have an ID. Set in the Document mapping (one line):

```
tpwrite "Dataset name: '[::T4X::TC::MAPPING::FieldMapping \
  "$ItemRevisionType:IMAN_specification:MSWord" "Name"]'"
```

In order to access all the Dataset's properties, set this preference value additionally:

```
IMAN_specification:MSWord:#__getAllProperties__#:Properties
```

3.4.6 Read Teamcenter Classification Attributes

The T4x mapping function provides also a possibility to read data from the Teamcenter classification. It is not necessary to have a classification in the EA system corresponding to the Teamcenter classification. T4x can read the Teamcenter classification data and store them in strings. It does not matter if these are then transferred to EA classification data or used in any other way.

T4x supports reading Teamcenter classification data including multiple classifications (i.e. an `Item` or `ItemRevision` is in more than one class). The following steps have to be done:

- Configuration of classification as a data source to read Teamcenter classification information (in Teamcenter preferences), e.g.:

```
<T4x>_<TargetObjectType>4ItemRevision=
IMAN_classification:icm0
items_tag:Item:IMAN_classification:icm0
```

You have to state the classification for Item and ItemRevision separately, if you need both of them.

- Definition of field mapping data (in the appropriate mapping file), see examples 1 to 6 in `<Read classification attributes - t4x_read_tc_classification_attributes_doc.sd>`. The Example 3 shows you how to create the `FieldMapping` call to read the classification attribute name which has been identified using the test mapping:

  ```
  TcData(ItemRevision :IMAN_classification:
  ClassID2:ClassAttributeName:-60005)=Base unit.
  ```

  T4x is also able to recognize empty classification attribute values. If you have, for example, the following values in such a multi value classification attribute: "1","","3","4" T4x will return one string with these four values separated by "new line characters" `\n` in order to see which of these values are empty.

3.5 Structure Transfer

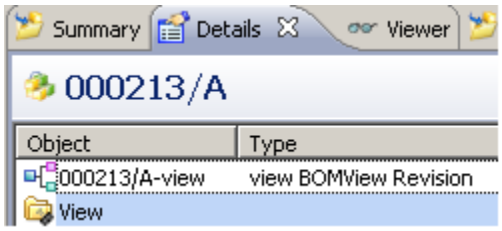
The Teamcenter BOM structure is read using a predefined revision rule. T4x sends always one level of a BOM to the EA . Sub assemblies have to be transferred in different transactions. T4x also allows transferring occurrence notes to BOM line specific data to the EA (e.g. the "bulk indicator", position text etc.). It is possible to distinguish between Item types, e.g. different Item types for design parts and raw materials. These might be transferred to set the BOM line e.g. as different position types in the EA.

Please note that:

- Bill of material (BOM) is only one possible TargetTypeName. You can choose to define your own TargetTypeNames to be able to transfer different configurations of BOMs.
- The expressions "BOM line" and "BOM position" are synonymous.
- The Generic BOM Mapping can be applied to all Teamcenter POM object types.
- You can configure T4x to extract data from several levels of the Teamcenter BOM structure. So you can add additional Items to be sent from a lower level. The number of levels sent to the Enterprise Application depends on your mapping logic only.

Caution:

Although in most cases T4x uses the complete Teamcenter object type name that is shown in the column "Type" (e.g. "view BOMView Revision" in the following screen shot), T4x uses the short type name as shown in the object name (here: "view") for the BOM handling (same in the preferences and the mapping):



The function `TC_Object2<EA>_Object` is executed once for the whole BOM, so here are the settings for the BOM header. Then the function `TC_Object2<EA>_Position` is called once for each BOM position.

Many EA, especially ERP systems, are able to distinguish between changes in the BOM that were made interactively and changes made by an automatic update mechanism like T4x. Depending on your business processes you may want to take this concept into account, e.g. by explicitly setting some special attribute for BOM positions (sometimes called CAD indicator or data sovereignty). However the solutions in different EA system vary widely, so no general solution can be provided here.

How to read "Global Alternate" Item data

Get the data of the "Preferred Global Alternate" Item of the current Item Revision:

```
$ItemRevisionType:items_tag:$ItemType:preferred_global_alt:$ItemType
```

Get the data of the "Global Alternate" Item of the current Item Revision:

```
$ItemRevisionType:items_tag:$ItemType:global_alt_list:$ItemType
```

Get the corresponding data for a stated AlternateID:

```
ItemRevision:altid_list:<AlternateRevisionIdType>
ItemRevision:items_tag:Item:altid_list:<AlternateIdType>
```

For more details about the structure transfer see chapter **Structure Transfer Specific Handling** and **Read Data of Structure**.

3.5.1 Structure Transfer Specific Handling

To define the BOM view types and data mapping from Teamcenter to the EA the following adaptations in the Teamcenter preferences have to be done:

- Definition of supported BOM view types.
- For each type the revision rule has to be defined.

The example defines 2 BOM view types (view and EAview). For both types the same revision rule is used (Latest Working). Further, you may set a list of valid BOMHeader ItemRevision types and define the data for processing from Teamcenter specifically:

```
<T4x>_<TargetStructureType>TypeList=
view
<EAview>

<T4x>_<TargetStructureType>Mapping4view =
Latest Working

<T4x>_<TargetStructureType>Mapping4<EA>view =
Latest Working

<T4x>_<TargetStructureType>HeaderTypeList =
<ItemRevision>

<T4x>_<TargetStructureType>HeaderMapping4<ItemRevision> =
items_tag:<Item>

<T4x>_<TargetStructureType>LineMapping4<ItemRevision> =
items_tag:<Item>

<T4x>_<TargetStructureType>OccurrenceNotes4view=
bl_has_children
bl_rev_view
```

Please note that:

- The BOM `TypeList` only refers to the BOM view types themselves, no matter which Item Type they belong to.
- You do not have to specify the data T4x has to read from Teamcenter for processing the BOM lines (similar to the preference `<T4x>_<TargetObjectName>Mapping4<TC_Object>`). As in most cases the BOM header and each BOM position are also used for an unstructured transfer (e.g. MaterialMaster or Item), T4x uses the information in `<T4x>_<TargetObjectName>Mapping4<TC_Object>` for reading Teamcenter data for object and structure transactions in the same way. If you have many entries in this preference (which means

T4x will try to read a lot of data from Teamcenter for the BOM header and every BOM position as well), you can reduce BOM processing time by reducing the amount of data to read for the BOM processing. Therefore, add the preference `<T4x>_BillOfMaterialLineMapping4<TC_Object>` with only the needed entries. Additionally, you may specify Item Revision Types for BOM processing that is not needed for e.g. materials. This is only needed in special cases, e.g. for using another Item Revision Type as a text position in the BOM. The advantage of using this preference is that you do not have to care about the additional type (i.e. you do not even have to specify it in any of the TypeList preferences). For example, using a Document Revision as a BOM position additionally, just set it for the BOM Mapping as follows:

```
<T4x>_<TargetStructureType>LineMapping4Document Revision=
IMAN_master_form:Document Revision Master
```

- For workflow transactions T4x needs to know which ItemRevision types are valid as BOMHeader types. The preference `<T4x>_<TargetStructureType>HeaderTypeList` is optional. If not set all ItemRevision types are valid for the transfer.

- The preference `<T4x>_<TargetStructureType>HeaderMapping4<TC_Object>` controls the amount of data that is processed for the BOMHeader object. To achieve best processing performance only add entries to the preference that are really required inside the mapping logic. The same applies for the preference `<T4x>_<TargetStructureType>HeaderMapping4<TC_Object>` that controls the data export for each BOMLine type. You need to add a preference for each ItemRevision Type that shall be transferred to the EA system.

> Caution:
>
> Although an ItemRevision Type may be not specified in `<T4x>_<TargetStructureType>LineMapping4<TC_Object>`, T4x will try to read it if it is used as a BOM position. In order to avoid strange effects, you should configure T4x in that way not to use a BOM position if it is of a special type (of course you can exclude positions for other reasons in the same way, see below). The following code in the BOM position mapping function `TC_Object2<EA>_ObjectPosition` will just skip the desired position without creating an error or stopping a Workflow Job (i.e. the transaction will end successfully and this position will just be missing in the EA as if it didn't exist in Teamcenter):
>
> ```
> if {$ItemType == "xy"} { tpwrite "Skip BOM pos $PosIndex because its
> Item Type is xy" return "SKIPPED" }
> ```

Occurrence Notes are small texts that may be attached to a Teamcenter BOM position in only this BOM. First you have to configure in Teamcenter preferences which T4x should read. T4x does not limit the number of Occurrence Notes, but be aware that this is an additional data transfer which decreases performance when processing large data. The following example shows the usage of the two Occurrence Notes `bl_has_children` (standard in Teamcenter) and `My_OccNote` (a custom type):

```
<T4x>_<TargetStructureType>OccurrenceNotes4view=
bl_has_children
My_OccNote
```

Caution:

T4x is only able to read Occurrence Notes of type `string` (check this in the BMIDE). If BOM positions are packed in the Teamcenter Structure Manager, Teamcenter will not give you the complete data on these positions: it will not tell the alternative positions, and if there are Occurrence Notes, it will only tell "PACKED NOTES", neither their names nor the values. If you want to transfer unpacked BOM lines, use the workflow argument `–unpack_all_bomlines=true`.

- Some Occurrence Notes are used by T4x by default without having to configure them explicitly, e.g. the `Quantity`.

- The Occurrence Note `bl_has_children` is set in a Teamcenter BOM if this has another BOM on its own, so you can determine a BOM structure in Teamcenter.

An Occurrence Note type has to be defined in the BMIDE.

In the next step you have to define the BOM data mapping to the EA. The function `TC_Object2<EA>_Object` is executed once for the whole BOM, so here are the settings for the BOM header (see above). Then the function `TC_Object2<EA>_Position` is called once for each BOM position.

The example shows the reverse mapping for two BOM view types to their Item Revision Master Forms. Note that the ItemRevision Master Form may be the same for both BOM view types, so only one preference "FieldMapping" is needed:

```
<T4x>_<TargetStructureType>Mapping2view=
structure_revisions:ItemRevision:IMAN_master_form:ItemRevision Master

<T4x>_<TargetStructureType>Mapping2ERPview=
structure_revisions:ItemRevision:IMAN_master_form:ItemRevision Master

<T4x>_<TargetStructureType>FieldMapping2structure_revisions:ItemRevision:
IMAN_master_form:ItemRevision Master=
item_comment
user_data_3
```

Caution:

- Be aware of the settings for different Item Types for BOM as described in the beginning of this chapter.
- The T4x Action-Handler `<T4x>-transfer-GenericObject` and Rule-Handler `<T4x>-validate-GenericObject` will need the argument `-TargetTypeName` with the value `BillOfMaterial` (or another custom value you define for your implementation) in order to use the relevant preferences and branch in the generic BOM mapping.
- Additionally the argument `-useView4Transfer` with the value `TRUE` and
- `-BomHeaderTypeList` e.g. with the value `Item`.
- For detailed information about the handler please see chapter **Process Based Trigger (Workflow)**.

Please note, that:

- In the BOM mapping, the BOM header has always the "PositionIndex" 0, whereas the BOM Positions have the numbers 1, 2...
- The "PositionIndex" is a parameter to the BOM mapping functions (BOM and BOM position mapping function as well). In the BOM position mapping function, the variable "BomLineIndex" is used additionally because of possible index changes.
- If a Teamcenter BOM contains alternative positions, T4x will transfer them all to the EA (which may be useful or not, depending on your processes). If you do not want that, you can prevent it in the mapping. As only the "preferred" alternative (the one you will see in Teamcenter Structure Manager) has a Find Number (or Sequence Number, respectively) in the Teamcenter BOM, you can evaluate that (see "Example 2"). As this is no problem from neither T4x nor the EA point of view, a BOM workflow job is not aborted by default and the user is not informed directly.

Example 1:

```
# Do not pass a BOM Position to the Enterprise Application if its
#   "ArticleNumber" (user_data_? In the samples) is empty
#   (if it is taken from a TC attribute that is filled when the
#   Article is created):
if {$ArticleNumber == ""} {
    tpwrite "BOM custom mapping does not use Position $PositionIndex:
       ArticleNumber not found (does Article exist in EA?)"
    return "SKIPPED"
}
```

Example 2:

```
# Do not pass a BOM Position to EA if its Position Number is empty
#    (read from the Find/Sequence Number; it might be an alternative
#     position):
if {$Position == ""} {
    tpwrite "BOM custom mapping does not use Position $PositionIndex:
       no TC BOM Sequence No (is it an alternative position?)"
    return "SKIPPED"
}
```

The examples below show data mapping of transaction time and function code to the Item Revision Master Form attribute "item_comment". All fields have to be configured in the Teamcenter preferences before (see above):

```
if {($Status eq "ERROR") || ($Status eq "UNKNOWN")} {
  ::T4X::TC::MAPPING::storeReverseMappingAttribute BomHeader \
  $ItemRevMaster item_comment "$TargetTypeName $BomViewType $Status -\
  [clock format [clock seconds] -format "%Y-%m-%d %R"]"
} else {
  ::T4X::TC::MAPPING::storeReverseMappingAttribute BomHeader \
  $ItemRevMaster item_comment "$TargetTypeName $BomViewType sent to \
  ERP - [clock format [clock seconds] -format "%Y-%m-%d %R"]"
}
```

For more details and more workflow handlers please see the API documentation.

3.5.2 Read Data of Structure

The function ::T4X::TC::MAPPING::IndexedFieldMapping is used to read the structured data from Teamcenter. To read the data from the header or top element in the structure, put the `"0"` for the Index. To get the data from elements iterate through the structure using variable `$PositionIndex` as it implemented in example below:

```
# Read some attributes from Teamcenter object from the header or top
level
  set PositionIndex 0
  set Id      [::T4X::TC::MAPPING::IndexedFieldMapping $PositionIndex\
   $ItemRevMaster user_data_1]
  set ItemId  [::T4X::TC::MAPPING::IndexedFieldMapping $PositionIndex\
   $Item item_id]

# Read some attributes from Teamcenter object from the variable
# PositionIndex, please set the wished index, for example 1
  set PositionIndex 1; # first line under the top level
  set Id             [::T4X::TC::MAPPING::IndexedFieldMapping \
    $PositionIndex $ItemRevMaster "user_data_1"];
  set ItemId           [::T4X::TC::MAPPING::IndexedFieldMapping \
```

```
    $PositionIndex $Item "item_id"]
  set Position       [::T4X::TC::MAPPING::IndexedFieldMapping \
    $PositionIndex BOMLine "bl_sequence_no"]
  set Quantity       [::T4X::TC::MAPPING::IndexedFieldMapping \
    $PositionIndex BOMLine "bl_quantity"]
```

3.5.3 Read more than one level of a Teamcenter BOM

T4x can send only one level of BOM data to the EA system by default. However T4x can read multiple levels of a Teamcenter BOM. A common use case is that a certain Teamcenter BOM line represents a structure level only and its children need to be included to EA Bill of Material.

So T4x can skip the structure BOM line and extract its children and send those to the EA.

The following description is valid for all T4x based transfers of BOMs. The example uses the sample Teamcenter item type `EA_T4x_MfgItem` and the custom occurrence note name `EA_T4xOccNote`, which must be replaced by the actual type and occurrence note used in your configuration. In all preference names, the prefix T4x must be replaced by the correct product prefix (e.g. T4S, T4O, T4EA...).

T4x offers three methods to identify BOM lines whose children need to be extracted as well. Additionally, you need to add more functionality to the BOM Position mapping. Within the T4x BOM workflow handlers you have to use the argument `-scan_max_bom_level` to define the number of levels eligible for export (the default value is 1; so increase the number to get lower levels at all).

You may also set this to "0" (zero), then T4x will not read any position of the current Teamcenter BOM but only the BOM header.

Option One: Use a Teamcenter preference to define the ItemRevisionType to use

The name of the preference is `<T4x>_<BomTargetType>LineGetNextLevelTypeList` and the value(s) are the relevant ItemRevisionTypes. The value has to be of type ItemRevision or Part Revision.

```
T4x_BillOfMaterialLineGetNextLevelTypeList= EA_T4x_MfgItem Revision
```

For all Teamcenter BOM lines of Item Revision type `EA_T4x_MfgItem Revision` the data of the children will be extracted from Teamcenter as well.

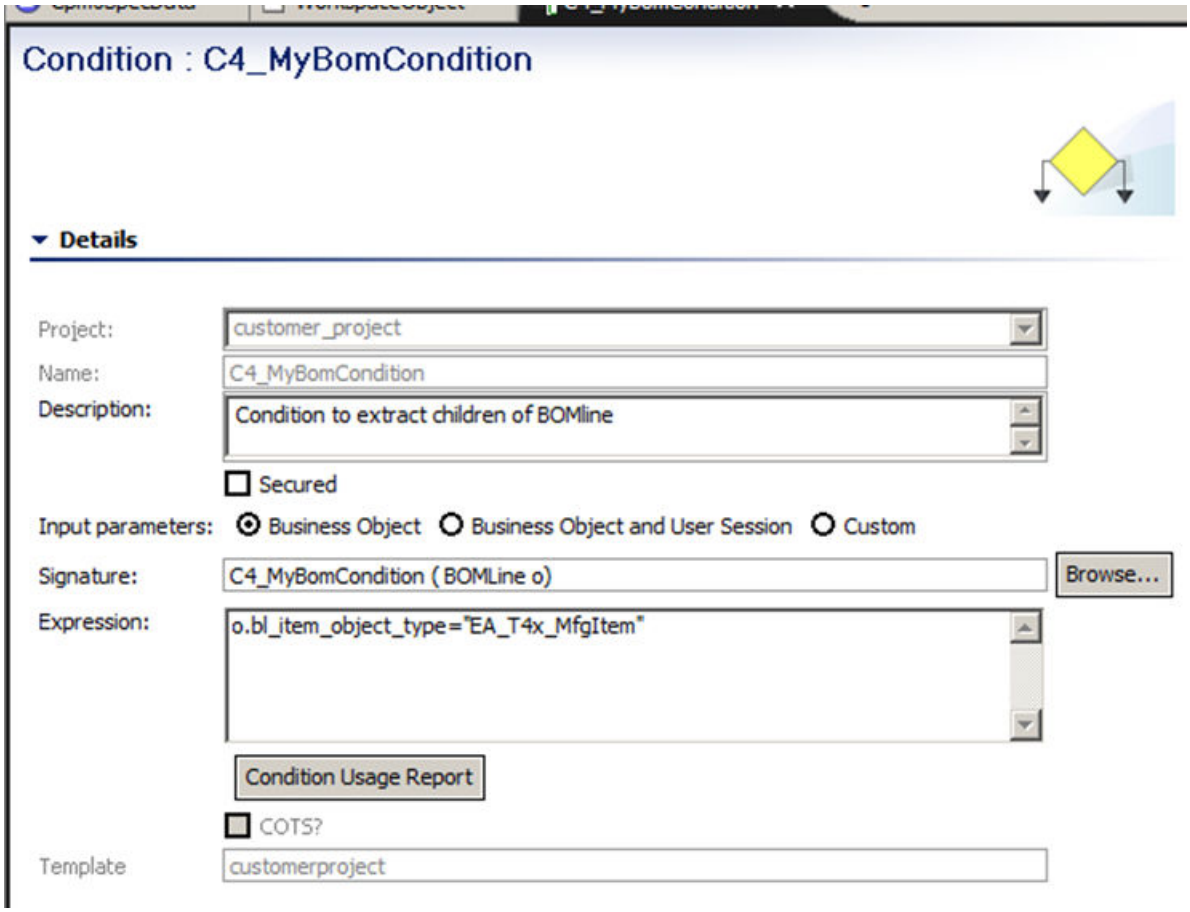
Example: output from BOM test mapping script including the lower level BOM lines:

```
TcData(2:EA_T4x_MfgItem Revision:IMAN_master_form_rev:EA_T4x_MfgItem
Revision
Master:object_name)=000009/A
TcData(2:EA_T4x_MfgItem Revision:IMAN_master_form_rev:EA_T4x_MfgItem
Revision
Master:object_tag)=QRF5LIWio8b5mBAAAAAAAAAAAAA
TcData(2:EA_T4x_MfgItem Revision:IMAN_master_form_rev:EA_T4x_MfgItem
Revision
```

```
Master:parent_object_tag)=QNO5LIWio8b5mBAAAAAAAAAAAAA
+ TcData(2:1:BOMLine:bl_has_substitutes)=
+ TcData(2:1:BOMLine:bl_item_item_id)=000008
+ TcData(2:1:BOMLine:bl_item_item_id)=000008
+ TcData(2:1:BOMLine:bl_quantity)=
+ TcData(2:1:BOMLine:bl_sequence_no)=15
```

The indexes 2:1 in the high-lighted lines indicate the stated object is the first sub-item of BOM position 2.

Option Two: Use a Teamcenter Condition to decide whether to load children



In BMIDE define a condition on BOM line level in order to decide whether the children of the BOM line should be loaded. Whenever the condition returns value true, its children will be extracted. When using conditions you can define a more complicated logic to make the decision making of sub-item extraction more powerful. See Teamcenter documentation on defining conditions on BMIDE.

Additionally, you need to tell T4x to use the condition by adding another preference:

```
T4X_BillOfMaterialLineGetNextLevelCondition =
C4_MyBomCondition
```

Option Three: Decide on BOM line level to load additional children

You need to set two preferences. First define the BOM line critical attribute, i.e. which Teamcenter BOM line occurrence note has to be checked:

```
<T4x>_<BomTargetType>LineGetNextLevelOccNote
```

Example:

```
T4x_BillOfMaterialLineGetNextLevelOccNote=
EA_T4xOccNote
```

Then define the BOM line occurrence note value that should initiate the loading of the children:

```
<T4x>_<BomTargetType>LineGetNextLevelOccNoteCondition=
CompareAction1
Value1
CompareAction2
Value2
...
CompareActionN
ValueN
```

<T4x>: Product Key (e.g T4S, T4O, ...)

<BomTargetType>: Traget BOM Type (e.g. BillOfMaterial, EquiBillOfMaterial, ...)

CompareAction:

- EQUAL: Use all the sub BOM lines that are equal the value defined next.
- NOT_EQUAL: Use all the sub BOM lines that are not equal value defined next.
- MATCH: Use all the sub BOM lines that match the value defined next.
- NOT_MATCH: Use all the sub BOM lines that don´t match the value defined next.
- RANGE: Use all sub BOM lines if the occurrence note value is an integer within a defined range. Define two integer values separated by comma, e.g. for "100,300" all values between 100 and 300 are valid.
- IN_LIST: checks if the occurrence value is found in the comma separated preference value list.

Example:

```
T4X_BillOfMaterialLineGetNextLevelOccNoteCondition=
EQUAL
EXPORT
```

With this configuration T4x will export lower level BOM lines for each BOM line that has the value `EXPORT` set on the occurrence note `EA_T4xOccNote`. BOM lines with different values will not be expanded.

See example in `<Read more than one level of a Teamcenter BOM – t4x_read_more_than_one_level_of_a_Teamcenter_BOM_doc.sd>`.

Please note that:

- T4x directly calls function `TC_Object2EA_BillOfMaterialPosition` only for the first level of Teamcenter BOM lines. All further sub BOM lines are handled in a recursive loop.
- The `PositionIndex` is structured as follows:
  - "0": top level of BOM (BOM header).
  - "1...n": level 1 BOM lines.
  - "1...n:1..n": level 2 BOM lines; e.g. "2:3" stands for the third BOM line on level 2 of following parent: second BOM line on level 1.
  - "1..n:1..n:1..n": level 3 BOM lines.

3.6 Error Handling

- **Process All Targets or One Transfer Type**.
- **Collect All Error Messages for One Transfer Type**.
- **Branch a Workflow in Case of an Error**.
- **Filter Already Transferred Objects**.

3.6.1 Process All Targets or One Transfer Type

- Handler argument `–collect_all_errors=true`

Pro	Con
Pop up can be used to inform user with error message	Depending on the number of targets and size of error message possible overflow of TC error stack (pop up may not contain all messages)
No further action is needed in workflow design	
All further workflow designs are possible (branching workflow etc.)	

- Handler argument `–continue_on_error=true`

Pro	Con
All further workflow designs are possible (branching workflow etc.)	Pop up can not be used to inform user with error message
	Validation task can not be used to branch workflow
	Further actions are needed in workflow design to respect transfer dependencies etc. (object must exist before structure transfer)

- Use a TC Sub-Workflows for each target object: Handler `EPM-create-sub-process` and -multiple_processes and –dependency=::

Pro	Con
No futher actions are needed in workflow design to respect transfer dependencies etc.	Deep Teamcenter design knowledge needed
All further workflow designs are possible (branching workflow etc.)	Object access to object in main workflow (change object) from sub workflow may not be possible
	Depending on the number of targets might be very resource consuming (100 targets = 100 workflows)

- Start export job(s) for each object and catch the result to trigger the workflow.

3.6.2 Collect All Error Messages for One Transfer Type

- Store error message to Teamcenter error stack (Handler argument `–collect_all_errors=true`).

Pro	Con
No access or write rights have to be granted to the target objects	Depending on the number of targets and size of error message possible overflow of TC error stack (pop up might not contain all messages)
No Teamcenter BMIDE changes needed	User has to filter or find the objects which have failed

Pro	Con
	No possibility to read the collected error message and use them for further actions, for example for e-mail messaging
	No possibility to filter already transferred objects during retry

- Store the message for each target to a defined property (directly on the target or Form related to the target object).

Pro	Con
Possibility to read the collected error message and use them for further actions, for example for e-mail messaging	Property on tartget object or related object has to be configured in BMIDE
Possibility to filter already transferred objects during retry	Access rights to the target object or related object has to be granted
	User has to filter or find the objects which have failed

- Store all messages for all targets to the T4x workflow dataset.

Pro	Con
No access or write rights have to be granted to the target objects	User has to filter or find the objects which have failed
No Teamcenter BMIDE changes needed	(Nearly) No possibility to read the collected error messages and use them for futher actions
	Access rights to the workflow dataset has to be granted
	Workflow dataset content is only partially configurable
	(Nearly) No possibility to filter already transferred objects during retry

- Store all messages for all targets to a Teamcenter workflow form and a defined property.

Pro	Con
No access or write rights have to be granted to the target objects	User has to filter or find the objects which have failed
Content of the form is completely configurable	Access rights to the workflow dataset has to be granted. But Form can be target of the workflow (workflow ACLs)

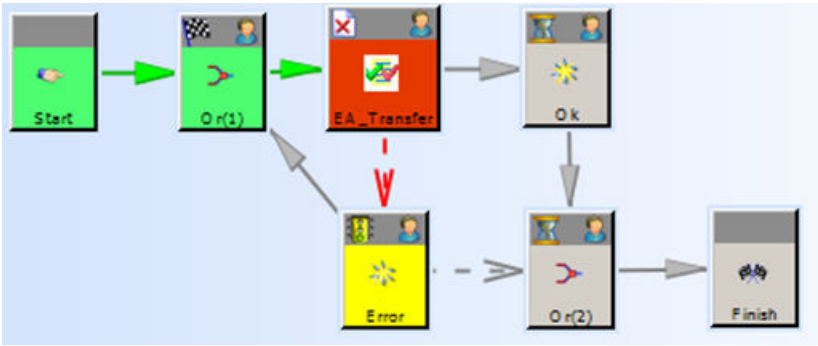
Pro	Con
Possibility to read the collected error messages and use them for futher actions	Teamcenter BMIDE configuration is needed
	No possibility to filter already transferred objects during retry

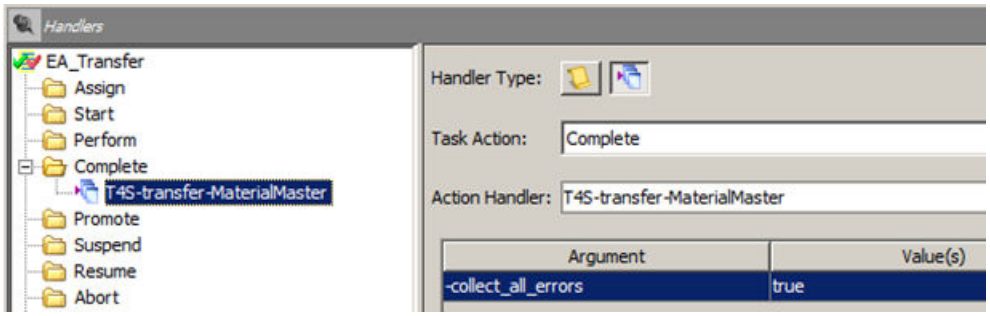
- Store all messages for all targets to an external storage (file, T4x shared mem, and so on).

Pro	Con
No access or write rights have to be granted to the target objects	User has to filter or find the objects which have failed
Possibility to read the collected error messages and use them for futher actions	Nearly no possibility to filter already transferred objects during retry (kind of complex)
No Teamcenter BMIDE changes needed	
No access rights changes needed	

3.6.3 Branch a Workflow in Case of an Error

- Use a validation task





Pro	Con
Easy workflow configuration	No direct user notification (pop up) possible
Could be extended to use multiple error codes and so use multiple different branches (generates a lot of effort)	
No TC (BMIDE) configuration needed	
Works for system errors like T4x not available and error which cause reverse mapping not to work, too	
Visual representation of OK and Error	

User defined error massages and error codes can be used to steer the workflow to different paths.
To use this functionality following configurations must be done:

- Define your messages in the language specific ue_errors.xml file: *$TC_ROOT/lang/textserver/<language>/ue_errors.xml*
  The error base needs to start by 919000, e.g.:
  `<error id=„1">User error message containing myParameter %1$</error>.`

- Define your error message in your mapping file with `storeUserMessage` function (supports up to 5 optional parameters), e.g.:

  ```
  ::T4X::CORE::storeUserMessage "ERROR" "919001" $myParameter.
  ```

  `return ERROR` mapping will return the code 919001, and this code can be validated via a validation Task in Teamncenter.

- Use a validation task, and add error paths. Assign error codes to the error paths. A single code or a list of codes can be assigned to a error path.

- Depending on which error code the mapping returns, the workflow will follow the pre-configured error path.

Validation task supports two options:

- Single Error Path: 1. OK Path (in case of no errors), 1. Error Path for all errors (no code defined, "any" option is selected).

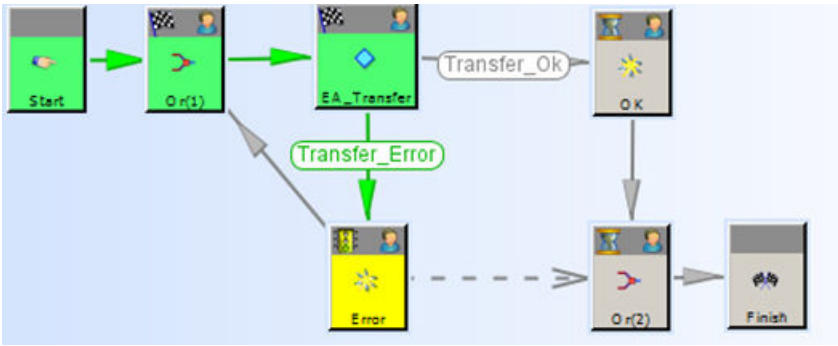
- Multiple Error Paths: 1. OK Path (in case of no errors), 1. Error Path (for codes for instance 919001 and 919002), 2. Error Path (for code 919003).

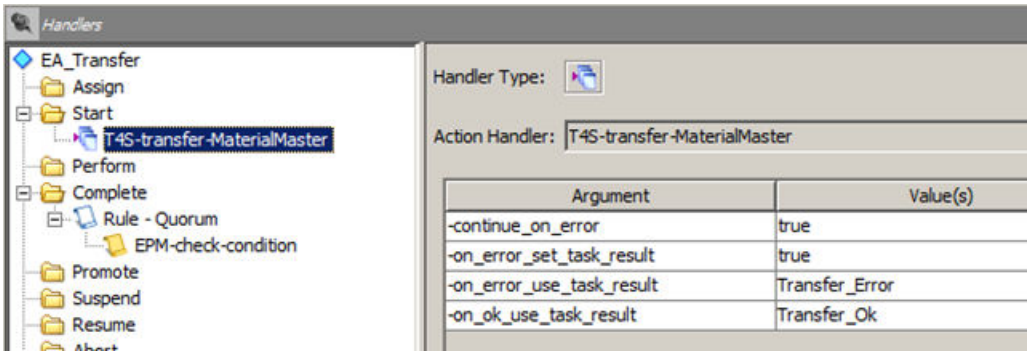
Caution:

Once you define an error code for a path, you must define codes for other error paths as well. This means, in case of the following option, the workflow will halt at the validation task and will not follow any paths:

1. OK Path, 1. Error Path (for codes 919001 and 919002), 2. Error Path (no code defined, "any" option is selected).

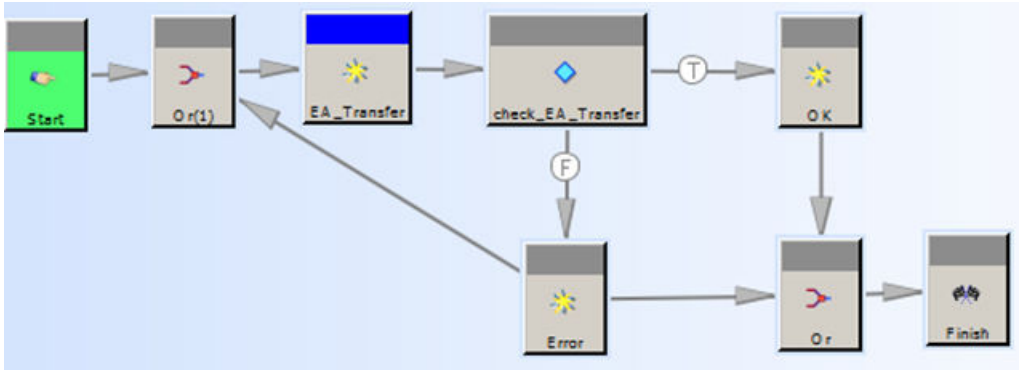
- Use a condition task and handler's argument

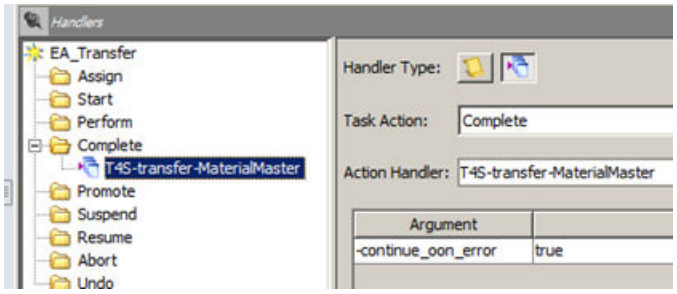




Pro	Con
Easy Workflow configuration	No direct user notification (pop up) possible
No Teamcenter (BMIDE) configuration needed	No visual representation of OK and Error
Works for errors (config errors) which cause reverse mapping not to work	Does not works for system errors like T4x not available

- Use a condition (or validation Task) and property which is filled during reverse mapping





Pro	Con
Works for errors (config errors) which cause reverse mapping not to work if the property which is checked is set by Teamcenter handler to "error value"	No direct user notification pop up possible
Works with Teamcenter < TC8.3	No real visual representation of OK and Error
	Does not works for system errors like T4x not available
	Additional workflow configuration needed. Condition `Task=Query` which evaluates property value etc.
	Teamcenter (BMIDE) configuration needed for reverse mapping property

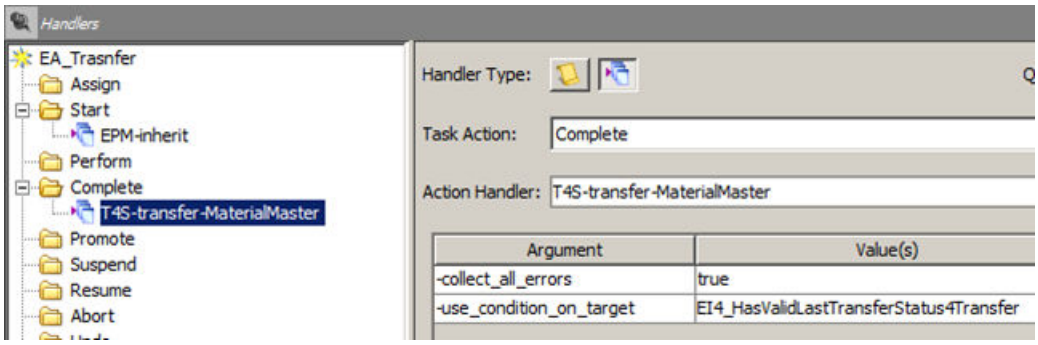
3.6.4 Filter Already Transferred Objects

- Store the transfer status to the target or related object and use mapping.

Pro	Con
Works also for properties which are not directly on the target object	BMIDE configuration needed for reverse mapping property
Can be quickly configured also in already running environments	TC data is still exported - performance
No additional BMIDE configuration needed	

- Store the transfer status to the target or related object and use BMIDE condition.

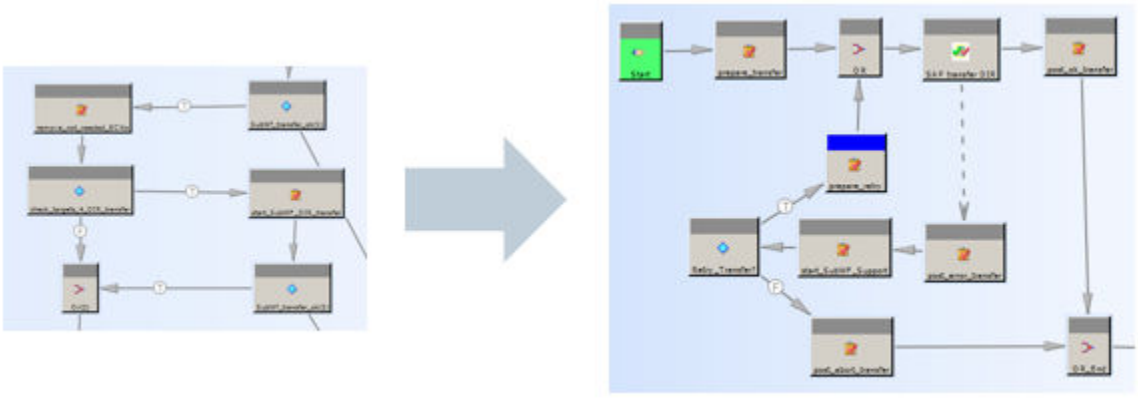
Pro	Con
Teamcenter data is not exported, performance improvement	Teamcenter BMIDE configuration needed for reverse mapping property
	Additional BMIDE configuration needed for condition
	Does not work on properties which are not directly on the target object - compound property on the target object is needed in this case
	BMIDE condition do not support differt object_type with (maybe) different properties. Property must be defined on a parent class of all objects.



- Start a sub workflows for transfers, store transfer status to the target or related object and remove successful transferred objects from the target `folder`.

Pro	Con
Teamcenter data is not exported, performance improvement	Teamcenter BMIDE configuration needed for reverse mapping property
No additional Teamcenter BMIDE configuration needed	Maybe new challenges in respect to process ownership, access objects / information from main workflow

Pro	Con
Does work on properties which are not directly on the target object (default `EPM-set-decision` is using a query)	Needs advanced knowledge about workflows etc.
	User do not see directly the transfer flow (has to navigate from sub to the main workflow)



3.7 Advanced Configuration

- Write an email from T4x
- Use of BMIDE Conditions (CLIPS) for Configuration of Rules
- How to read properties from a Teamcenter relation
- How to change the ID and name of all Teamcenter objects in the current Item Revision during a T4x transfer
- How to get detailed Teamcenter Release Status information
- Set the Teamcenter Release Status effectivity dates
- Attach a Teamcenter form to the current workflow job and read it via the Teamcenter Gateway
- Read and write Teamcenter table properties
- Attach more functionality to a workflow
- How to ignore read issues during data export
- Preference Configuration for Derived Types

3.7.1 Write an email from T4x

The T4x Job Server has a functionality included to send e-mails on certain conditions, e.g. a failed job, see the **Job Alerts** for details.

Besides that, T4x allows sending an email from within an Admin UI script or in the mapping. This may be especially useful if a Teamcenter user should be notified when he cannot get an error message immediately, e.g. his input is needed in a failed Teamcenter Workflow job that has been started by another user. Furthermore the T4x e-mail functionality allows notifying several users at the same time, not only the one who started the Workflow job.

Example for sending the same mail with two file attachments to three different users:

```
set mailfrom  name@company.com
set mailto    "user1@company.com user2@company.com user3@company.com"
set smtp      $SMTP_Host ;# machine name or IP address of your SMTP
server
set subject   "T4x status mail"
set mailbody  "T4x sent this test e-mail"
set org       "T4x at [info hostname]"
set xmailer   TEST
set mailformat     TEXT ;# TEXT, HTML
set mailprio       no_prio ;# high, low, any
set attachments "C:/temp/otto.txt C:/temp/willi.txt"
set rc [::T4X::CORE::sendMail2BGS2 $mailto $mailfrom $smtp \
  0 $mailbody $org $subject $xmailer $mailformat $mailprio $attachments]
tpwrite "--- sendMail2BGS2 returns $rc ---"
```

Please note that:

- The mail may be sent in TEXT or HTML format, to be specified as parameter `mailformat`.
- In order to send the mail with a specific priority, set the parameter `mailprio` to `high` or `low`, respectively; with any other value (e.g. the value `no_prio` in the example above) T4x will not try and give a priority to this e-mail.
- This allows sending several attachments. The whole mail size is limited by the SMTP server only, not by T4x.
- As shown in the example above, use a slash as directory separator (no backslash), same in Windows and UNIX.
- No matter if the mail was sent, the return value will be `0` (zero) in most cases.
- As indicated by the function name `sendMail2BGS2`, the mail is always sent by the BGS, never by T4x Apps, so the only physical machine that needs access to the SMTP server is the BGS host, no Teamcenter or T4x client.

- SMTP servers usually do not return proper error messages.
- If the mail is not sent, first check the BGS log file:
  - BGS Admin UI – Applications – System logs.
  - Click through *sys/<machine_name>/<BGS_instance_name>/tpbgs64.log*.
  - It should show some lines with `::EMail::` that should contain a hint to the actual problem.
- If this does not help, ask your IT administration to check network connections and about how to address the SMTP server; especially the allowed values for Org and Xmailer.
- Besides that, there is another function to send an e-mail directly from the tpapps. In principle it works the same from the user's point of view but it does not allow all the same parameters, but only the following ones:

```
set rc [::EMail::sendmail2 $mailto $mailfrom $smtp \
  0 $mailbody $org $subject $xmailer]
```

3.7.2 Use of BMIDE Conditions (CLIPS) for Configuration of Rules

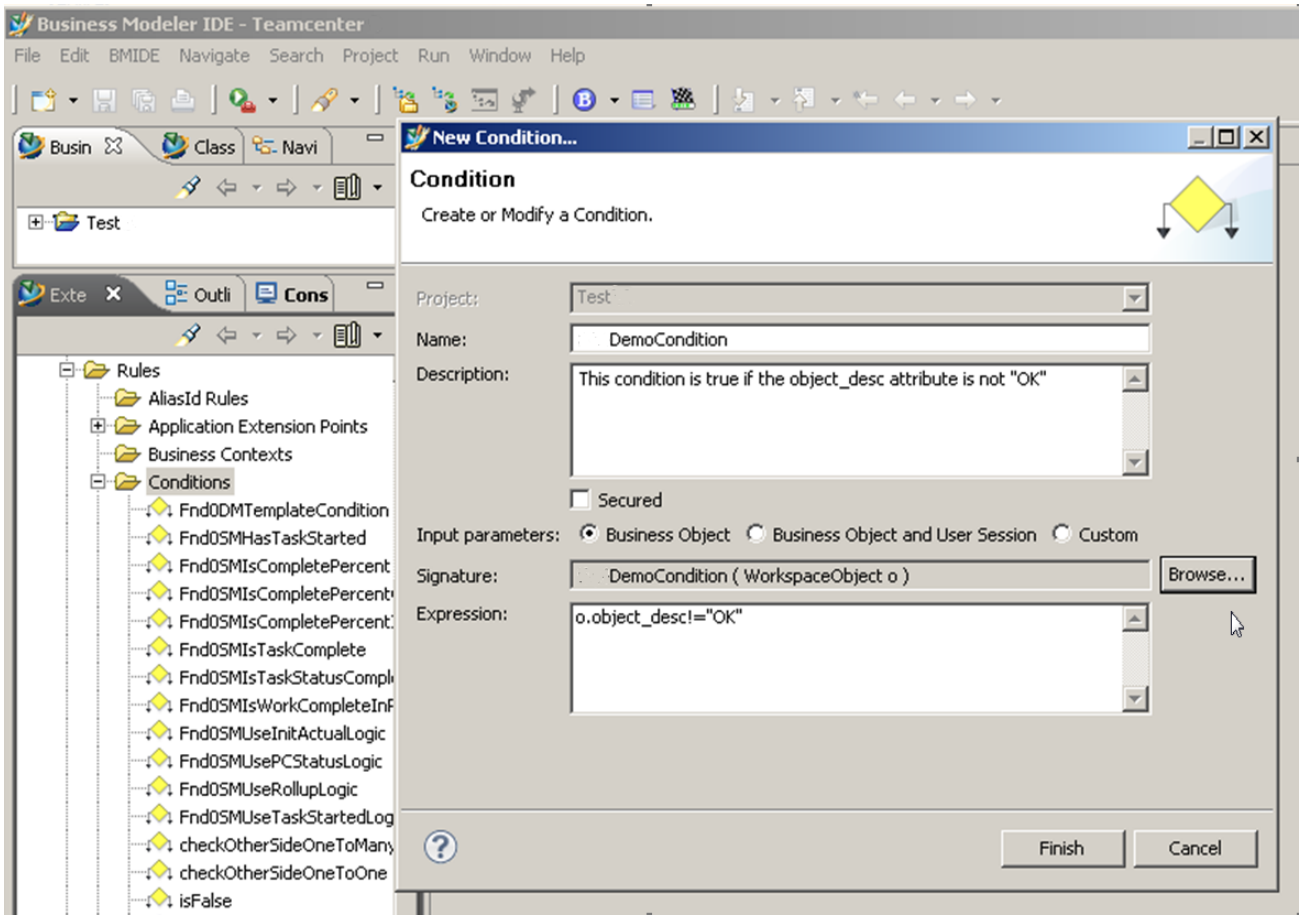
Teamcenter allows defining special conditions in BMIDE that may be used with T4x in order to allow a possibility to control the transfer of a workflow target object without programming. This OOTB solution may replace customer specific Workflow handlers and therefore facilitate the Workflow handling. Using this functionality T4x will check if a Workflow target has to be processed or not without checking the mapping code and therefore reduce the needed overall Workflow processing time.

The functionality using the BMIDE based conditions is only active if a specific workflow handler argument is specified. It is supported for `<T4x>-transfer-GenericObject` and `<T4x>-transfer-GenericBOM`.

For the support of the transfer conditions you need to define a BMIDE condition (see screen shot below) and this condition needs to be defined with the following signature:

`<ConditionName>` (WorkspaceObject o)

(e.g. DemoCondition (WorkspaceObject o))



The code in the condition should return `TRUE` if the condition is fulfilled and can call any other sub condition that is needed. More details on how to define conditions and sub conditions can be found in the BMIDE online help.

In a second step an additional handler argument needs to be specified in the used transfer handler to enable the support of the BMIDE condition during the transfer. All these handlers (see above) now support a new handler argument to enable the BMIDE condition based transfer:

```
-skip_bomlines_by_condition=true
```

The validated conditions are defined in a site preference with the following naming convention:

```
<ProductKey>_<TargetTypeName>SkipCondition4<BOMLineType>
```

This does not need additional workflow handler arguments or mapping code.

3.7.3 How to read properties from a Teamcenter relation

Since the Teamcenter version 10 it is possible to define properties directly on the relation objects. That way data can be stored only in a specific context. T4x supports reading those attributes.

You need to extend the use case specific preference to read the relation attributes. Then you can access the property from the mapping.

Scenario: On material master transfer, we determine the relevant SAP target plant from Teamcenter plant object, e.g. `SAP2_MEPlant` that is related to the material part with the relation `SAP2_PlantUsage`. The SAP plant ID is equal to the Plant object's Item ID, the plant specific material status is derived from the relation property `sap2_plant_status`. In that way you can store different plant specific status values within Teamcenter and transfer them separately to each SAP target plant.

Preference:

```
T4S_MaterialMasterMapping4SAP2_T4S_Item Revision =

SAP2_PlantUsage:SAP2_MEPlant:#__getAllProperties__#:Properties
```

Mapping:

```
set Plant [::T4X::TC::MAPPING::FieldMapping "$ItemRevisionType: \
 SAP2_PlantUsage:SAP2_MEPlant" item_id]
set PlantStatus [::T4X::TC::MAPPING::FieldMapping "$ItemRevisionType: \
 SAP2_PlantUsage:SAP2_MEPlant" "RelationProperty:sap2_plant_status"]
```

Whenever you define the key `#__getAllProperties__#:Properties` within the preferences to get *all* properties for the given object, it is strongly recommended to use either the Blacklist or Whitelist feature to limit the amount of data that is processed. See **Blacklist/Whitelist**.

3.7.4 How to change the ID and name of all Teamcenter objects in the current Item Revision during a T4x transfer

T4x may change the ID and name (`object_id` and `object_name`) of the Teamcenter Item and all its Item Revisions and Datasets in the reverse mapping, i.e. in the part of the transaction T4x executes after modifying the corresponding EA object. A common use case is changing the object IDs to the EA object key such as the material number instead of writing this number into a form attribute in Teamcenter. Together with the Item ID, this functionality can change the IDs of all Datasets in the current Item Revision or even in all the Item Revisions in the current Item.

The following preferences are used for that functionality:

```
<T4x>_<TargretObjectName>FieldMapping2Item= AllIdStrings object_desc
```

Set the value `AllIdStrings` to activate that "rename all" functionality for the object type Item, preferably as the first value in this preference. Of course you have to activate the reverse mapping for the Item, i.e. set the Item in the `Mapping2` preference of the current object type, e.g.:

```
<T4x>_<TargretObjectName>Mapping2ItemRevision= items_tag:Item
```

Then T4x will do that renaming for all the Item Revisions and all their Datasets in the current Item.

In order to only rename the Item itself, the current Item Revision and its Datasets, but not the other Item Revisions in that Item and all their Datasets, set those two preferences as follows (of course they may have more values):

```
<T4x>_<TargretObjectName>Mapping2ItemRevision= ItemRevision
<T4x>_<TargretObjectName>FieldMapping2ItemRevision= AllIdStrings \
object_desc <T4x>_ItemRenameStopReverseMappingIfItemIdExists
```

If this optional preference is set (the value does not matter), T4x will stop the whole rename process if the Item ID does already exist in Teamcenter. This is the reason why the value for the Item should be the first in the `Mapping2` preference and the value `AllIdStrings` should be the first in the `FieldMapping2` preference then no reverse mapping action is done before that renaming. So if the renaming has to be stopped because the desired Item ID already exists, no other reverse mapping will be done in the same transaction (because that could be wrong then).

`<T4x>_ItemRenameSeparator4ItemId` This optional single value preference defines a character (e. g. "-") which is searched in the existing object IDs. If this preference is set, i.e. such a separator character is defined, the rename action will only be performed for the part of the ID/name before this character.

Example: the value of this preference is "-" and the existing Item ID is 1234567-890, then only the "1234567" will be renamed and the last part "-890" remains unchanged.

`<T4x>_ItemRenameDatasetDescription` If this optional preference is set (the value does not matter) the dataset description will be changed additionally.

`<T4x>_ItemRenameDatasetRule= MSWord:Ignore UGMASTER:Name` This multi-value preference is mandatory for that functionality; it defines how the Datasets should be handled during that renaming. The values must follow the rule `<Datasettypename>:<ReplaceType>` where the `<Datasettypename>` is the Teamcenter Dataset type name and `<ReplaceType>` is either `Name` or `Ignore`. Datasets of type MS Word will not be renamed at all and every UGMASTER will get the same new name as the Item itself. Every Dataset type that is not listed in that preference (e.g. DirectModel) will get the same new ID as the Item.

> Caution:
>
> The `FieldMapping2` preference differs from the usual way: although it is needed to use both the expressions `item_id` and `object_name` in the mapping, none of them has to be set as a preference value. As shown above, only the preference value `AllIdStrings` has to be set for that functionality: `<T4x>_MaterialMasterFieldMapping2Item= AllIdStrings`.

Example:

```
set TC_ NewObjectName [::T4X::TC::MAPPING::FieldMapping $Item
object_name]
::T4X::TC::MAPPING::storeReverseMappingAttribute MaterialMaster $Item \
```

```
    AllIdStrings $NewObjectId ;# rename the Teamcenter Item
  ::T4X::TC::MAPPING::storeReverseMappingAttribute MaterialMaster $Item \
    AllNames $NewObjectName ;# always use AllNames and AllIdStrings
```

Both lines are needed in the mapping, regardless if you like to change only the `object_id` or `object_name`. If you want to keep the old value, you need to read it from Teamcenter first and then set the variable to the old value.

3.7.5 How to get detailed Teamcenter Release Status information

By default, T4x reads the basic data of a Teamcenter Release Status only, e.g. the name and the date when it was added (which is not the same as the valid date!). In order to get more detailed information about all the last Teamcenter Release Status of an object, add the following value to the corresponding "Mapping4" preference:

```
<T4x>_<TargetObjectName>Mapping4ItemRevision=
last_release_status:ReleaseStatus
```

Then you will find some more data about the Release Statuses, e.g. the `NumberOfEffectivities` and `object_tag` of this status object. Note that setting this additional preference value only allows reading the Teamcenter Release Status information additionally and this special functionality is not included in the preference value `#__getAllProperties__#:Properties .`

3.7.6 Set the Teamcenter Release Status effectivity dates

Add the following preference values:

- ```
  <T4x>_<TargetObjectName>Mapping2ItemRevision=
  last_release_status:ReleaseStatus
  ```

- ```
  <T4x>_<TargetObjectName>Mapping2last_release_status:ReleaseStatus=
  Dates:0:0
  Dates:0:1
  ```

In the mapping set something like this:

- ```
  set ItemRevStatus "$ItemRevisionType:last_release_status:ReleaseStatus"
  ```

- ```
  ::T4X::TC::MAPPING::storeReverseMappingAttribute \
    Article $ItemRevStatus "Dates:0:0" $VALID_FROM
  ```

- ```
  ::T4X::TC::MAPPING::storeReverseMappingAttribute \
    Article $ItemRevStatus "Dates:0:1" $VALID_TO
  ```

You may leave out the line with `$VALID_TO`, then it will be set to open end (till infinity) by default. In most cases, only the last release status and its first effectivity should be updated. If there is none, a new
  ```

effectivity will be created. In order to handle another effectivity than the first one, modify the first zero in `Dates:0:0`, e.g. `Dates:2:0` for the "valid from date" of the third effectivity. In this case, be sure to handle Teamcenter's restrictions regarding the different dates correctly.

3.7.7 Read and write Teamcenter table properties

Reading Teamcenter table properties in the mapping

If the extension `#__getAllProperties__#:Properties` is used to export data from the object, T4x will also export the data of the table property. If this is not required, use the black- or whitelist functionality to decrease the amount of data.

To read each column, the field name parameter of the `FieldMapping` function needs to be extended with the column number (e.g. `MyTableRow:1`).

Example:

```
TcData(ItemRevision:MyTableRow:1)=First column
TcData(ItemRevision:MyTableRow:2)=Second column

set TableRowpropety MyTableRow
set NumberOfTableRows     [::T4X::TC::MAPPING::FieldMapping \
  $ItemRevision "$TableRowProperty:NumberOfRows"]
for {set i 0} {$i < NumberOfRows} {incr i} {
  set MyColumn1 \
   [::T4X::TC::MAPPING::FieldMapping $ItemRevision \
   "$TableRowProperty:$i:MyColumn1"]
  set MyColumn2 \
   [::T4X::TC::MAPPING::FieldMapping $ItemRevision \
   "$TableRowProperty:$i:MyColumn2"]
}
```

Writing Teamcenter table properties in the reverse mapping

The reverse mapping function will only update the table as a whole. That means within the mapping you need to define the whole content of the table and then write it back.

Unlike the function `::T4X::TC::MAPPING::storeReverseMapping`, the procedure is more complicated and includes several steps.

The first step for creation update or deletion of the content of the property is to set the TableType and TableProperty as defined in the BMIDE and initiate the mapping buffer:

```
# table type as defined in the data model
set TableType     CustomTableType
set TableProperty CustomTableProperty
#
```

```
# create internal mapping buffer to store the table row properties
#
::T4X::TC::MAPPING::createReverseMappingTableObject $Relation
$TableProperty \
   $TableType
```

If you want to delete the content of the table property, then you can stop here. If you want to keep the existing data you have to read out the property and initialize the existing rows:

```
#
# copy the existing table rows to mapping buffer without changing
# the row properties
#
set NumberOfTableRows [::T4X::TC::MAPPING::FieldMapping $Relation \
   "$TableProperty:NumberOfRows"]
for {set i 0} {$i < $NumberOfTableRows} {incr i} {
  set RowTag1 \
  [::T4X::TC::MAPPING::FieldMapping $Relation \
    "$TableProperty:$i:object_tag"]
  set TableRow \
  [::T4X::TC::MAPPING::addReverseMappingTableObjectRow \
   $Relation $TableProperty $RowTag1]
  # optional: read the content of the table row to identify the
  # RowTag for change
  set RowContent1 \
  [::T4X::TC::MAPPING::FieldMapping $Relation "$TableProperty:$i"]
}
```

If you want to update an existing table row, you will need the RowTag of that row. You can identify the RowTag in the function above by its content. Then can change each column of the row separately.

```
 #
 # update only those row propties of a given
 # specific table row that needs to be updated
 #
 ::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2PlantId 9905
 ::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPType ND
 ::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPGroup 011
```

If you want to update an existing table row, you will need the RowTag of that row. You can identify the RowTag in the function above by its content. Then can change each column of the row separately.

```
#
# update only those row propties of a given specific table
# row that needs to be updated
```

```
#
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2PlantId 9905
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPType ND
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPGroup 011
```

To add new table rows use NULLTAG instead of a table row tag. Repeat this for each new row.

```
#
# add an new row to the buffer, in this case a non-existing one
#
set TableRow \
[::T4X::TC::MAPPING::addReverseMappingTableObjectRow $Relation \
  $TableProperty NULLTAG]
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2PlantId 9901
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPType ND
::T4X::TC::MAPPING::storeReverseMappingTableRowObjectAttribute \
  $Relation $TableProperty $TableRow sap2MRPGroup 012
```

After the the Reversemapping is completed the information will be stored to Teamcenter.

Reading and writing table properties with ITK

The following functions are provided to read and write table properties via ITK. Please look them up in the Teamcenter Gateway API Reference:

- `::ITK::AOM_ask_table_rows,`
- `::ITK::AOM_set_table_rows,`
- `::ITK::AOM_delete_table_rows.`

3.7.8 How to ignore read issues during data export

In certain use cases Teamcenter integration functions are used to traverse Teamcenter object trees where objects belong to different users and have different access rights for the different users. Traditionally these functions stopped execution with an error message, if read access for an object in the tree was not granted for the current user. With Teamcenter Gateway 18.1 we introduce a new preference to change the traversal behavior.

Functionality will be enabled by the side preference *T4X_enableReadAccessCheck* with the following importable XML description:

```
<preferences version="10.0">
  <category name="Gateway Foundation">
    <preference name="T4X_enableReadAccessCheck" type="Logical"
array="false" disabled="false" protectionScope="Site" envEnabled="false">
    <preference_description>This preference enables or disables the
"ReadAccess" privilege check during the Teamcenter data extraction.
                            If enabled (true) all object which have no
READ privilege will be skipped during the data extraction.
                            if disabled (false) the check will be
skipped and an "Access denied" error will be raised.

                            Default values is false.
    </preference_description>
      <context name="Teamcenter">
        <value>false</value>
      </context>
    </preference>
  </category>
</preferences>
```

If the preference does not exist or is set to *false*, the behavior of the T4X functions is still the same as before, i. e. object tree traversal will stop with an error message if read access for an object in the tree is not granted for the current user.

If the preference is set to *true* however, no read access errors will be raised and the traversal result will contain the data of the accessible objects.

3.7.9 Preference Configuration for Derived Types

Parent Teamcenter Business Object Type based data extraction configuration

The parent Teamcenter Business Object Type based data extraction is a way to simplify the preference definition during an implementation project. This approach supports the following preferences used by the Teamcenter Gateway data extraction functionality:

- Values of the "TypeList" preference: `<T4x>_<TargetTypeName>TypeList`. Example: `T4EA_ExternalObjectTypeList= $ItemRevision`
  This will enable the TargetTypeName "ExternalObject" for all types deriving from ItemRevision and ItemRevision itself.

- Values of the "Mapping4" preference: `<T4x>_<TargetTypeName>Mapping4<TcObjectType>`
  The syntax `$<Parent Business Object Type>` can be used instead of the individual Sub-Business Object Types in any level of the preference line:`<Relation Or Reference>:$<Parent Business Object Type>`

- Type in the "Mapping4" preference:
  `<T4x>_<TargetTypeName>Mapping4$<SelectedParentBusinessObjectType>` (only available for the following parent object types: "$ItemRevision", "$Dataset")

- Type in the "HeaderMapping4" preference:
  `<T4x>_<TargetTypeName>HeaderMapping4$ItemRevision` (only available for the following parent object types: "$ItemRevision", "$Dataset")

- Type in the "LineMapping4" preference:
  `<T4x>_<TargetTypeName>LineMapping4$ItemRevision` (only available for the following parent object types: "$ItemRevision", "$Dataset")

The following use cases are not yet supported:

- Type in the "Mapping4" preference:
  `<T4x>_<TargetTypeName>Mapping4$<ParentBusinessObjectType>` (for types other than ItemRevision or Dataset)

- Type in the "HeaderMapping4" preference:
  `<T4x>_<TargetTypeName>HeaderMapping4$<ParentBusinessObjectType>` (for types other than ItemRevision or Dataset)

- Type in the "LineMapping4" preference:
  `<T4x>_<TargetTypeName>LineMapping4$<ParentBusinessObjectType>` (for types other than ItemRevision or Dataset)

- Values in the reverse "Mapping2" preference:
  `<T4x>_<TargetTypeName>Mapping2<TcObjectType>`

- Type in the reverse "Mapping2" preference:
  `<T4x>_<TargetTypeName>Mapping2$<ParentBusinessObjectType>`

Sample Use Case

Lets assume there is a need to transfer the information of all related documents of an ItemRevision.

In this scenario you could specify the extraction paths via the following lines in the corresponding `*Mapping4ItemRevision` preference by specifying every individual Teamcenter Business Object Type:

```
IMAN_specification:Specification Revision
IMAN_specification:RequirementSpec Revision
IMAN_specification:EmailRevision
IMAN_specification:DocumentRevision
IMAN_specification:Specification Revision:items_tag:Specification
IMAN_specification:RequirementSpec Revision:items_tag:RequirementSpec
```

```
IMAN_specification:EmailRevision:items_tag:Email
IMAN_specification:DocumentRevision:items_tag:Document
```

Alternatively you can use the *Parent Teamcenter Business Object Type* syntax in the preference definition:

```
IMAN_specification:$DocumentRevision
IMAN_specification:$DocumentRevision:items_tag:$Document
```

The result of the data extraction is that all business objects which have the specified parent Teamcenter Business Object Type will be extracted.

Mapping Syntax

In the mapping you still need to use the *individual Teamcenter Business Object Type* with exact type names if you use the standard `::T4X::TC::MAPPING::FieldMapping` function. In addition there are the following new mapping functions that could be used:

- `::T4X::TC::MAPPING::searchObjectList`
- `::T4X::TC::MAPPING::searchRootTaskObjectList`
- `::T4X::TC::MAPPING::searchIndexedObjectList`

These procedures are supporting the wildcard in the parameter `FormNamePattern` and return a list of the found for FormName (relations) and the corresponding object tag. This can be used directly as input to the `::T4X::TC::MAPPING::getObjectAttributeValue` function. See the API guide for details.

Benefits

- The Parent Teamcenter Business Object Type functionality simplifies the way to configure how the data is extracted.
- Data extraction performance is improved, because less preference values have to be evaluated.
- If a new sub Teamcenter Business Object Type is added to the Teamcenter data model there is no need to change or extend the preferences.
- It is possible to deliver a neutral preference set as a starting point for a customer implementation.

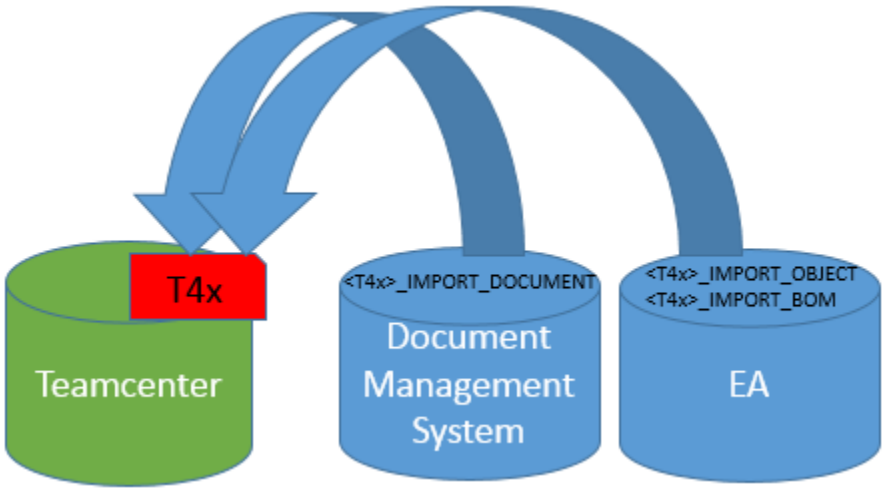
Downside

- You may get more data than needed, if you use *Parent Teamcenter Business Object Type* syntax and the data-model is not well adjusted to this situation.

4. T4x Import Concepts

In the T4x context Import means the import of data from the EA into Teamcenter. You can import structured and unstructured data and files.

There is a concept for typifying the imports similar to the transfer's TargetTypeName. There are two import types available in the standard configuration: `<T4X>_IMPORT_OBJECT` and `<T4X>_IMPORT_STRUCTURE`. If you want to use more descriptive names or need more separate import types, you can define your own import types using the TCL procedure `::T4X::BATCHJOB::IMPORT::CreateImportCodeProxy`, which takes the name of the import type (sometimes also called "function") as its first argument. This allows separating different import types in different mapping files.



For imports the data to be imported are extremely important. There are two possible ways to feed the data into the mapping:

1. The mapping is called only with an external ID and requests the data by calling the EA once it was started, or

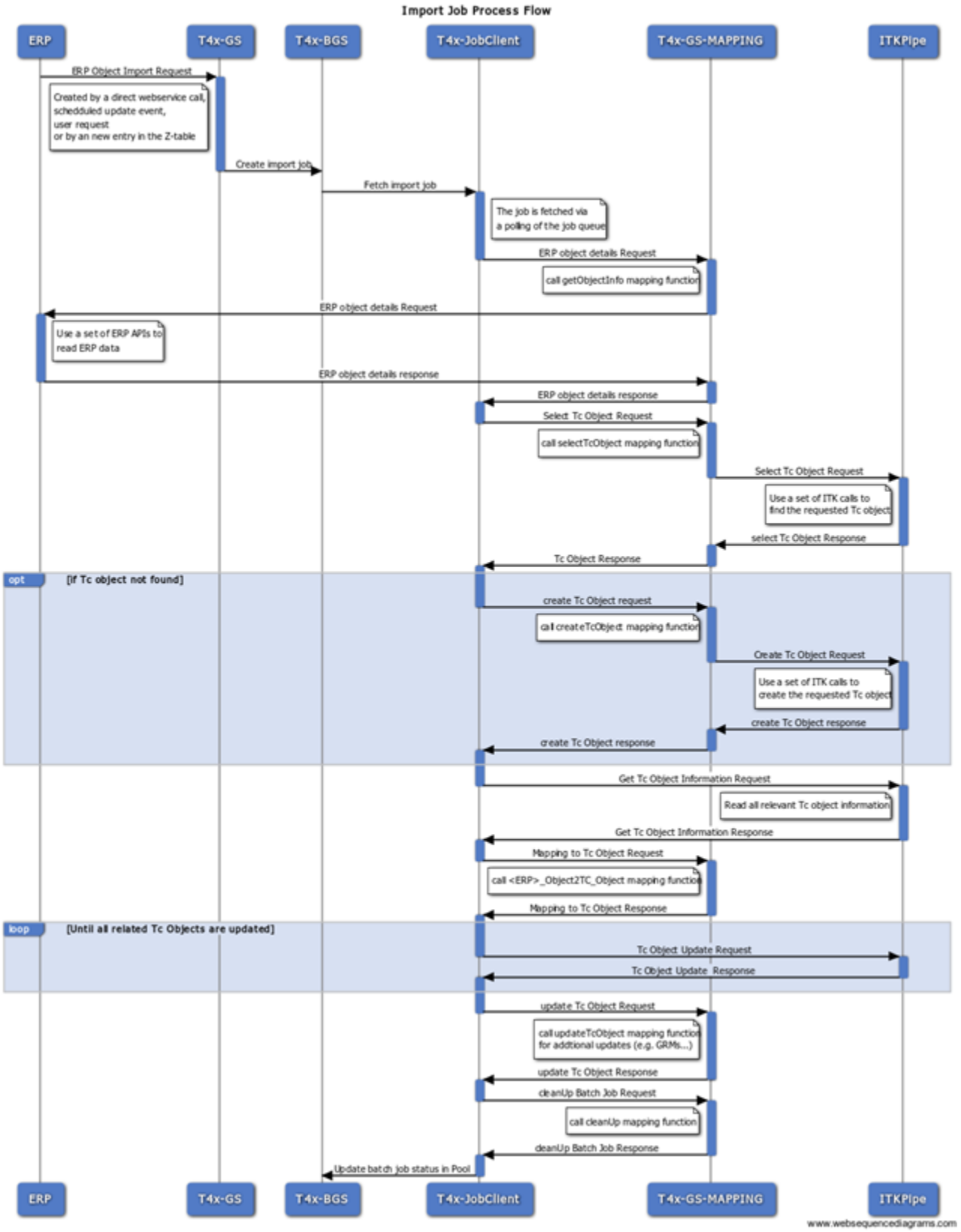
2. A complete set of data is fed into the initiating call by the caller using the `AdditionalInfo` parameter.

The first solution requires implementation of the data (e.g. a web service call, or a JDBC query) in the mapping. The second solution requires that the caller builds a plain TCL list with key/value pairs, passes this list as argument to the import call (AdditionalInfo) and the mapping evaluates this list. Note that in addition to the different implementation needs, there is an implication to the time the data is retrieved: Passing data as a list means the mapping works with data that was up-to-date at the time the import was triggered (or even before that, depending on the source). Having the mapping retrieve the data at the time of execution means that data is retrieved later, possibly significantly after the transfer was triggered, especially if the execution is deferred to a T4x job. Please keep this in mind, making sure that the data is indeed what it is meant to be at the time it is retrieved.

Usually imports identify the target object in the `selectTC_Object` procedure which needs to be implemented in the mappings. If the select procedure indicates that the requested data does not (yet) exist, the `createTC_Object` function will be called next, so a new object can be created using `::ITK` procedures. For updating existing Teamcenter data, imports may either use `::ITK` functions or use the so-called reverse mapping procedures
like `::T4X::TC::MAPPING::storeReverseMappingAttribute`.

The custom import procedures are embedded in a sequence of actions, which include several active instances. Figure above shows these steps for an import job in a generic way. The following parties are involved:

- EA: The ERP system or Enterprise Application.
- GS: The T4x Gateway Service.
- Job Server: The T4x Basic Gateway Service (BGS).
- Job client: The active component on the GS instance, which executes jobs.
- GS mapping: The custom part of the GS, the mapping (separated here to show the custom mapping procedures on a separate line. In reality the mapping is not executed in a separate active instance, but within the GS).
- ITK-pipe: Since there is no active Teamcenter user session, GS has to start a separate child process, connected to Teamcenter, executing all Teamcenter API Calls (ITK).



The different import mapping templates follow the same steps in general with one exception, the META Import mapping (details on this see below)

1. Get data from related EA object
2. Select corresponding Teamcenter object (either use TC query or selects with ITK calls – TC object type depends of job type, i.e. ItemRevision, Item, view, dataset)
3. Create missing Teamcenter object (optional)
4. Update attributes on Teamcenter object (also attributes on related objects)
5. Update Teamcenter object (trigger workflow,set status...)
6. Cleanup

Switch to the next chapter and learn more about the import concept:

- **Object Import**.
- **Structure Import**.
- **Import Advanced Configuration**.

4.1 Object Import

The following functions are used to import data from an EA into Teamcenter. It is usually triggered by a web service call or script. The EA import uses the OOTB import type `<T4x>_IMPORT_<EA_OBJECT>`. In some of the mapping templates you have to define you own import type since the OOTB type is already used. This definition is done by the call to `::T4X::BATCHJOB::IMPORT::CreateImportCodeProxy` with the import type `<T4x>_IMPORT_<YOUR_OWN_OBJECT>` at the beginning of the file.

`getObjectInfo`: is intended to retrieve data from the external system (via web services or JDBC query) and to transform data into an internal representation better suited for further processing. The resulting values are stored in a TCL dictionary named `ERPOutputDat` that is shared among the following procedures of this mapping.

`selectTC_Object`: tries to find a Teamcenter object that matches the input. The implementation may choose to find the Teamcenter object in any appropriate way. It could use a Teamcenter saved query to find the object or the procedure uses Teamcenter ITK functions to find the object with the given ID. If a matching object is found, this object will be updated, otherwise `selectTC_Object` returns an error and the framework additionally calls the procedure create`TC_Object`, which creates the requested item. After `selectTC_Object` (and optionally `createTC_Object`) have finished, the mapping has identified a target item with which the following functions can work.

`getTargetRevisionFromList`: This procedure is not required by the template but gets called by the specific `selectTC_Object` procedure. In case more than a single revision matches the criteria in `selectTC_Object`, this procedure selects the correct one from the given list.

`createTC_Object`: Creates an item of a given type, if no match can be found in Teamcenter by `selectTC_Object`. Note that you can influence the item type, name and ID in this procedure.

`GenObjMapping2TC_Object`: The procedure maps the attributes from the internal representation in `ERPOutputDat` to Teamcenter attributes. For some attributes, type conversions may be necessary (e.g. date formats of Teamcenter and XML schema differs). Finally the procedure sets the description to indicate the import date. The procedure does not directly write to Teamcenter but instead uses the framework procedure `::T4X::TC::MAPPING::storeReverseMappingAttribute` to store the values to be written to Teamcenter later on.

`updateTC_Object`: This procedure may optionally modify the Teamcenter object in a way that cannot be handled by the attribute mapping in procedure `GenObjMapping2TC_Object`. For example the Teamcenter "Unit of Measure" of the item can be set directly here.

`getERPOutputData` and `etXMLData`: convenience functions that return the value for the given key, if it exists and writes a warning to the log file and returns an empty string otherwise. These are not required by the template.

`cleanup`: Gets called at the end of the import, regardless of the previous status of the import. Should be used to clean up some resources (e.g. external files, open handles, global and namespace variables, arrays, etc.). Here only the `ERPOutputDat` dictionary is unset.

`verifyZTableIdStatus` and `SetZPTC_Status`: verifies and sets status in Z-Table.

4.2 Structure Import

The following functions are used to import/update a structure from an EA into Teamcenter. The import can be triggered by the script or web service call.

The OOTB import type `<T4x>_IMPORT_BOM` is defined by the call `::T4X::BATCHJOB::IMPORT::CreateImportCodeProxy` at the beginning of the file *var/mmap/<t4x>_mapping_config/<t4x>_bom_item_import_template.sd*.

`checkItemExists`: a convenience procedure not directly called from the framework. Checks if an item with the given ID exists in Teamcenter.

`getImportBomInfo`: gets the information on the BOM from the external source, in this case from the `AdditionalInfo` parameter. This procedure is not called from the framework but only from the custom mapping.

`getCurrentTcBomInfo`: gets the BOM information for the Teamcenter item revision. This procedure is not called from the framework but only from the custom mapping.

`importBOM2TcBOM`: integrate the Teamcenter BOM and the external BOM information into Teamcenter. At first the Teamcenter BOM and the external BOM are compared and no longer needed BOM lines are removed from Teamcenter BOM. In a second step existing BOM lines are updated, missing BOM lines are created.

This is quite a complex algorithm, which might be time intensive for large BOMs. Its execution time grows linear with the number of BOM lines. Under no circumstances you should change the algorithm to use a loop over one BOM structure within a loop over the other BOM structure (loop in loop) as the execution time of such a construct might explode (quadratic growth) with the number of BOM lines. See the Wikipedia article on "Analysis of algorithms" gives an introduction to this topic.

`getTargetRevisionFromList`: This procedure is not required by the T4x template but gets called by the specific implementation of the `selectTC_Object` procedure. In case more than a single revision matches the criteria in `selectTC_Object`, this procedure selects the correct one from the given list.

`selectTC_Object`: tries to find a TC object that matches the input. The implementation may choose to find the TC object in any appropriate way. It could use a Teamcenter saved query to find the object or the procedure uses Teamcenter ITK functions to find the object with the given ID. The current implementation always selects the latest revision for header and BOM items. If a matching object is found, this object will be updated, otherwise `selectTC_Object` returns an error and the framework additionally calls the procedure `createTC_Object`, which creates the requested item. After `selectTC_Object` (and optionally `createTC_Object`) have finished, the mapping has identified a target item with which the following functions can work.

`createTC_Object`: Creates an item of a given type, if no match can be found in Teamcenter by `selectTC_Object`. Note that you can influence the item type, name and ID in this procedure.

`selectTCBOMview_Object`: tries to find the BOM view revision that matches the input. If a matching object is found, this object will be updated, otherwise `selectTCBOMview_Object` returns an error and the framework additionally calls the procedure `createTCBOMview_Object`, which creates the requested BOM view.

`createTCBOMview_Object`: creates the corresponding target BOM view revision for the given item revision if no match can be found in Teamcenter by `selectTCBOMview_Object`.

`updateTC_Object`: This procedure can be used to do some additional customer specific post-actions after the normal update done in `importBOM2TcBOM`.

`cleanup`: Gets called at the end of the import, regardless of the previous status of the import. It should be used to clean up some resources (e.g. external files, open handles, global and namespace variables, arrays, etc.). Here only the Status is set to OK as no resources were acquired.

`verifyZTableIdStatus` and `SetZPTC_Status`: verifies and sets status in Z-Table.

Go to the chapter **Precise / Imprecise** or **Set a specific Teamcenter UoM** to learn more about how to configure the import mapping.

4.2.1 Set a specific Teamcenter UoM

By default, Teamcenter creates the BOM lines without a special UoM (unit of measure), no matter if the Item itself has a special one or the default value "each". In order to define the BOM line UoM during the Teamcenter BOM creation by the BOM import functionality, set the following section in the BOM import mapping file where the following line is originally:

```
::ITK::BOMLINE_addbomline_occurrence_notes_with_window \
$BomViewTag $TcWindowTag $ItemRevisionTag \
bl_sequence_no $SapPositionNo bl_quantity $bl_quantity
```

That means, comment that line out and add the following section directly after it:

```
# ::ITK::BOMLINE_addbomline_occurrence_notes_with_window \
                $BomViewTag $TcWindowTag $ItemRevisionTag \
                bl_sequence_no $SapPositionNo bl_quantity $bl_quantity

  set sUom KG  ;# The required BOM line UoM
  tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] -mtype DEBUG \
    "+++++ Try setting the TC BOM line UoM '$sUom' +++++"
  set sUomTagx [::ITK::UOM_find_by_symbol $sUom]
  tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] -mtype DEBUG \
    "+++++ ::ITK::UOM_find_by_symbol returns '$sUomTagx' +++++"
  if {[lindex $sUomTagx 0] == 0 && [lindex $sUomTagx 1] ne ""} {
    set sUomTag         [lindex $sUomTagx 1]
    ::ITK::BOMLINE_addbomline_occurrence_notes_with_window $BomViewTag \
    $TcWindowTag $ItemRevisionTag bl_sequence_no $SapPositionNo \
    bl_uom $sUomTag bl_quantity $bl_quantity
  } else {
    tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] -mtype DEBUG
\
      "+++++ Error: Teamcenter does not answer its internal UoM tag for \
       the UoM '$sUom'! +++++"
  }
```

Please note that by default Teamcenter does not allow setting the actual UoM value directly, but only the internal tag to the requested value, so that has to be checked before, which shown in the following example, too. The example above shows setting "KG" fix as the BOM line UoM value; maybe read it from the ERP system BOM which is imported instead of setting a fix value.

4.2.2 Precise / Imprecise

Teamcenter has an concept of precise and imprecise structure. You can easily change an assembly from precise to imprecise or back again, if it has not been released (releasing the assembly is a change to the structure). You may want to do this when the assembly reaches a different stage in the product life cycle, for example, when it is released from engineering to manufacturing. These considerations depend on your business process, and whether it is necessary to record precise references at any stage. If you

need to store the link whether to items or to item revisions for the structures, BOM view revisions, you have to execute following ITK-function on the parent line to set the tag on the structure to true (to set to precise, like in example) or false:

```
ITK::BOM_line_set_precise $HexBomLineTag true
```

By default, the BOM import will create a precise view in Teamcenter. However it can be changed to imprecise during the creation and update process in the mapping. To change the BOM view to imprecise, the following needs to be added to the function `getCurrentTcBomInfo` in the BOM import mapping to the part where the BOM header is initialized:

```
::ITK::BOM_line_set_precise HexBomLineTag OnOffFlag
```

Example:

```
proc getCurrentTcBomInfo {BomViewTag AdditionalInfo args} {
  tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] -mtype INTERN \
    "::T4S::BOM::CUSTOM::ITEM::IMPORT::getCurrentTcBomInfo Start"
  set Status "UNKNOWN"
  set TcBom  {}
  #
  # Initialize Bom Header Info
  #
  set BomDataStatus [::ITK::getObjectData $BomViewTag \
                     T4S BOM BillOfMaterial MaterialMaster]
  if {[llength $BomDataStatus] == 2 && [lindex $BomDataStatus 0] == 0} {
    ::T4X::TC::MAPPING::printIndexTcData TEST_MAPPING
    if {[llength [array names ::TcData *:ItemInfo]] > 0} {
      set Status "OK"
      set rc \
       [::ITK::BOM_line_set_precise $::TcData(0:ItemInfo:BomLineTag)
"0"]
      # "1" for precise or "0" for imprecise.
  […]
```

Caution:

- To change the whole BOM to imprecise, the BOM header line needs to be changed and not the BOM position lines.
- The HexBomLineTag of the BOM header is not the same as the BomViewTag or BomWindowTag.

4.3 Import Advanced Configuration

- **Teamcenter Multi-field Key Functionality**.
- **Write to Teamcenter Classification Attributes**.

4.3.1 Teamcenter Multi-field Key Functionality

The Teamcenter Multi-Field-Key functionality (also known with the abbreviation MFK) exists since Teamcenter 10.0; its main point is that more than one object may have the same ID. By default, i.e. if the MFK is not configured in Teamcenter, no object can be created with an ID that already exists for another object. The easiest way is that an Item of another type, e.g. Item, Part, Design, Drawing, (not possible for "sub types" like Dataset), can be created with an already existing ID, but there may be more different points in Teamcenter internally.

The MFK object definitions are defined via BMIDE, so maybe check in BMIDE of the currently used Teamcenter how it handles it.

With the main T4x transactions (e.g. Display, Create Direct, any default T4x workflow) the MFK functionality means nothing because then the required Teamcenter object has to be selected and therefore it does not matter if it has the same ID as another object. But if T4x should create a new Teamcenter object it must handle the MFK functionality because else Teamcenter may not allow creating the new object. The following T4x functions help with the MFK functionality:

By default, the import mapping templates only use the Item ID for unique object identification. If you need to enable MFK support, you have to do the following adaptions in the mapping functions (see above which functions are used):

- `selectTC_Object` Append a list of additional attribute name and value pairs to the call of `::T4X::CUSTOM::MAPPING::TOOLBOX::selectTC_Object`. For example, change the line from:
  `[::T4X::CUSTOM::MAPPING::TOOLBOX::selectTC_Object LATEST item_id $MatNumber]`
  to
  `[::T4X::CUSTOM::MAPPING::TOOLBOX::selectTC_Object LATEST item_id $MaterialNumber object_desc [::T4S::TC::MAPPING::SAPFieldMapping MaterialMaster $MaterialNumber MATERIALDESCRIPTION:MATL_DESC:1]]`
  similar function in order to use by T4x a Teamcenter object that is not selected but only stated with its Item ID and additional needed data, e.g. the Item Type:

```
set ItkStatus [::ITK::ITEM_find_item_revs_by_key_attributes \
 $ItemRevisionId item_id $ObjectId]

if {[lindex $ItkStatus 0] == 0 && [lindex $ItkStatus 1] == 1} {

  set ItemRevisionTag [lindex [lindex $ItkStatus 2] 0]

}
```

  If at least one corresponding Teamcenter ItemRev is found, then the function returns the list of ITK_ok (=0), the number of found revisions and the found Item Revision tags; else a list of another number than zero, which means error, and an error message. Similar function to search the Item no matter with which Item Revision (same result):

```
set ItkStatus [::ITK::ITEM_find_items_by_key_attributes item_id
$ObjectId]
```

- `createTC_Object` Append a list of additional attribute name and value pairs to the call of `::T4X::CUSTOM::MAPPING::TOOLBOX::createTC_Object`. For example, change the line from:

  ```
  set Status [::T4X::CUSTOM::MAPPING::TOOLBOX::createTC_Object \
   $NewItemId $NewItemName $NewItemRevisionId $NewItemType]
  ```

  to:

  ```
  set Status [::T4X::CUSTOM::MAPPING::TOOLBOX::createTC_Object \
   $NewItemId $NewItemName $NewItemRevisionId $NewItemType object_desc

  [::T4S::TC::MAPPING::SAPFieldMapping MaterialMaster $MaterialNumber
  MATERIALDESCRIPTION:MATL_DESC:1]]
  ```

  Example for a standard Item (Item type = Item) with the new Item ID 12345:

  ```
  set Status [::T4X::CUSTOM::MAPPING::TOOLBOX::createTC_Object \
   12345 ItemName12345 000 Item object_desc "new Item 12345"]
  ```

4.3.2 Write to Teamcenter Classification Attributes

Attributes and classes in the Teamcenter classification are identified by their ID, since their names are not unique. The relation between the Item and the class is represented by a so-called ICO. The T4x reverse mapping function can write to the Teamcenter classification as well as to other Teamcenter data. The following object reverse mapping example (for MaterialMaster) sets the attribute with ID 1000 in the class ICM01 to the value "Test1000" for the current `ItemRevision` and the value "Test1000" to the attribute with ID 1001:

```
::T4X::TC::MAPPING::storeReverseMappingAttribute MaterialMaster
  "$ItemRevisionType:IMAN_classification:ICM01" 1000 Test1000
::T4X::TC::MAPPING::storeReverseMappingAttribute MaterialMaster
  "$ItemRevisionType:IMAN_classification:ICM01" 1001 Test1001
```

In earlier versions, T4x could write to the Teamcenter classification with the following code sequence only. Now it is not necessary any more to do it like that but it still works. The following code sets the attribute with ID `$attributeId` in the class `$classId` to the value `$value` for the Item `$ItemTag`.

If the item is already classified, nothing will be changed. The code can be used in the reverse mapping `updateTC_Object`. The value of `$ClassStatus` is `OK` in case everything worked fine and contains the error code and information in case something went wrong. Please learn more about mapping in `<Write to Teamcenter Classification Attributes - t4x_write_to_tc_classification_attributes_doc.sd>`.

5. Job Processing Concept

T4x provides the following job script types:

- **Transfer Job Scripts:**
  These scripts are useful for testing the complete T4x configuration and mapping, as they process the data (same as the mapping test scripts) and send them to the EA as well as the answer of the EA system back to Teamcenter. In principle, their handling is the same as for the mapping test scripts. When applying this test script in "Direct" mode it is the same as a "Create direct" (or "Change direct", respectively) transaction from the T4x GUI on this object. Alternatively you have the option to execute it in "Send to Job Pool" mode for creating a job for the Create/Update transaction (transaction waiting to be executed by a T4x server in the background). For more details about the usage of these test scripts, please see chapter **Using Create Jobs on the Job Server** .
  Note that in context with jobs, the denominations "export" and "transfer" are partially handled synonymously.
  Set the *Category* filter in the *Script – Scripts* screen of the Admin UI to "Export" to filter for transfer job scripts.

- **Import Job Scripts:**
  These scripts transfer ERP objects to corresponding Teamcenter objects (Items or new Item Revisions of existing Items). For import mapping settings they use their own mapping files: `<t4x>_<object_type>_import_template.sd`. To be able executing those, you need to "source" the corresponding *.sd* file in *<t4x>_mapping_config.sd*, e.g.:

  ```
  source -relax t4s_mm_item_import_template.sd
  source -relax t4o_item_import_template.sd
  source -relax t4ea_item_import_ofbiz.sd
  ```

- Using the corresponding file allows adapting the behavior of this script in the same way as in all the T4x mapping files. As with transfer job scripts you can decide to execute the import job scripts either in "Direct" mode or in "Send to Job Pool" mode.
  Set the *Category* filter in the *Script – Scripts* screen of the Admin UI to "Import" to filter for import job scripts.

For more information about jobs and job management, please consult the following chapter:

- **Configuring Job Processing**
- **Using Create Jobs on the Job Server**
- **Job Pool Functions Script on Job Server**
- **Job Alerts**
- **Job Server Hints and Troubleshooting**

5.1 Using Create Jobs on the Job Server

The previous chapters explain how to configure the Job Server initially as well as how to work with the Job Server on a daily basis, e.g. to check jobs and their results. This chapter explains how to configure AIG, Teamcenter and the EA system in order to use the AIG Job Server in a way that fulfills the desired data handling.

> Caution:
>
> If the Job Pool is full, i.e. it already has as many jobs stored as the configured "Maximum number of jobs" then a new job will not be created immediately. Therefore maybe reset the Job Pool, see division "Reset Job Pool" in **Job Pool Functions Script on Job Server**.

Create Jobs by AIG scripts

Creating the simplest possible job has been explained in Admin UI help. It tells the BGS to create a job and execute it in stand-alone mode. Such a job does not need to handle network connections between different machines and does not try to connect to Teamcenter or the EA. It is just for checking if the internal job handling works correctly.

The next test should be executing the transfer test script. Every "job" script (no matter if executed in "Direct" or "Send to Job Pool" mode) needs the BATCH EA auto login, even if a mapping test script reports:

```
+++ INFO: Connection to <EA_name> is OK
```

This may be done as follows in the file *<t4x>_mapping_config.sd*. As ITK connection is needed, be sure to start this GS with the correct environment (`set TC_DATA`...):

```
::ITK::setCredentialsAlias <CredentialsAlias>
::T4X::CONNECTION2EA::setCredentialsAlias4UseCase <product>
<destination> <credentials_alias> BATCH
::T4X::CONNECTION2EA::selectActiveConnection2EA <product> * <destination>
```

Please try the script with the default mode "Direct" first, i.e. AIG will not use the Job Server functionality but directly start the transaction. If this finished with "OK" execute it again with the mode "Send to Job Pool" in order to involve the Job Server. Only a few seconds after clicking "execute", you should find the job in the state "Ready" in the "Job Server Job List" (see help topic in Admin UI). After it passed the "Running" state (i.e. it has been processed), check the "Logging".

The message `Batchjob ::T4X::BATCHJOB::EXPORT::createBatchjob finished with OK!` in the "Debug" output does not necessarily mean that the job was *processed* successfully, but only that it was *created* successfully. As the script finishes after the job creation is finished, it does not even notice the processing and cannot state its result therefore.

Caution:

In case of an error the job will return to status "Ready" again in order to retry the processing after the "retry time" which is one hour by default (see list of the "BatchjobDefaults" above). So you might think it is not executed at all. Please check the "Logging" in order to find the error reason which is in most cases a missing Teamcenter or ERP login.

Create Jobs from within the T4x mapping

In order to trigger specific actions depending on some data found during a transaction, it may be useful to create a job from within the mapping. This is what the mapping function `createBatchjob` does. Depending on the need to create an import job (get data from somewhere and import something to Teamcenter, i.e. create/update a Teamcenter object depending on the data found) or an export job (get data from Teamcenter and export them to somewhere else), there are two different T4x mapping functions with the same name `createBatchjob` but in different namespaces (for further details please see *API Reference* delivered with this manual). The correct function signatures are:

```
proc ::T4X::BATCHJOB::EXPORT::createBatchjob BatchJobName FunctionName
ObjectTag ObjectClassName ObjectType
ObjectId ObjectRevisionId {TransferDirection TC_EXPORT} {UserServices 0}
{SessionId BATCH} {Filter NONE} {TargetTypeName NONE} {DependOnIds {}}
args
```

```
proc ::T4X::BATCHJOB::IMPORT::createBatchjob FunctionName ImportObjectId

{TransferDirection TC_IMPORT} {ZTabId NONE} {AdditionalInfo NONE}
{SessionId BATCH} {Filter NONE} {DependOnIds {}} args
```

Create Jobs triggered by a Teamcenter Workflow

After a Teamcenter workflow job is started, the Teamcenter user is blocked as long as the workflow job is running locally on the Teamcenter user's machine. This is standard Teamcenter behavior. As AIG may process a lot of transactions in one workflow job, this may take a while.

In order to not block the Teamcenter user during this time, the AIG Job Server can be configured to process the workflow in the background so that the user can continue using Teamcenter right after launching the workflow. This is achieved by using the Teamcenter standard functionality to hand the workflow job over to another Teamcenter user. If this other user is not a human but the AIG Job Server, this processing can be done immediately without other Teamcenter users interference and without disturbing anybody's work.

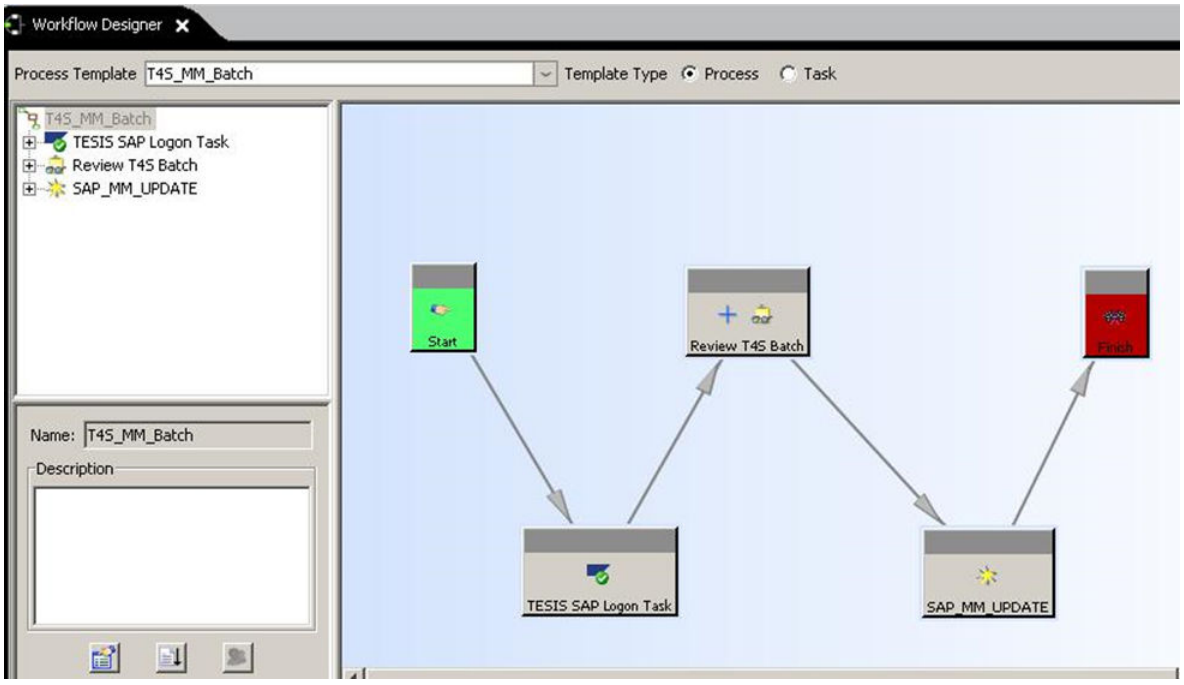
The process flow is as follows:

- The Teamcenter user starts a workflow job including AIG handlers.

- This workflow is assigned to another Teamcenter user, e.g. "t4s_batch" / "t4o_batch" / "t4xbatch".
- On another machine where no human needs to be logged in, T4x creates a connection to Teamcenter as this "t4s_batch" / "t4o_batch" / "t4xbatch" and to the ERP system.
- As there is no human to type the passwords on this other machine, this needs to be done automatically (see above):
  - ITK auto login set fix to the Teamcenter user "t4s_batch"/"t4o_batch"/"t4xbatch".
  - ERP auto login may be set to a fix user as well or
  - the interactive Teamcenter user who started the workflow logs in to the ERP system before and this existing login is then used by the AIG Job Server for this job.

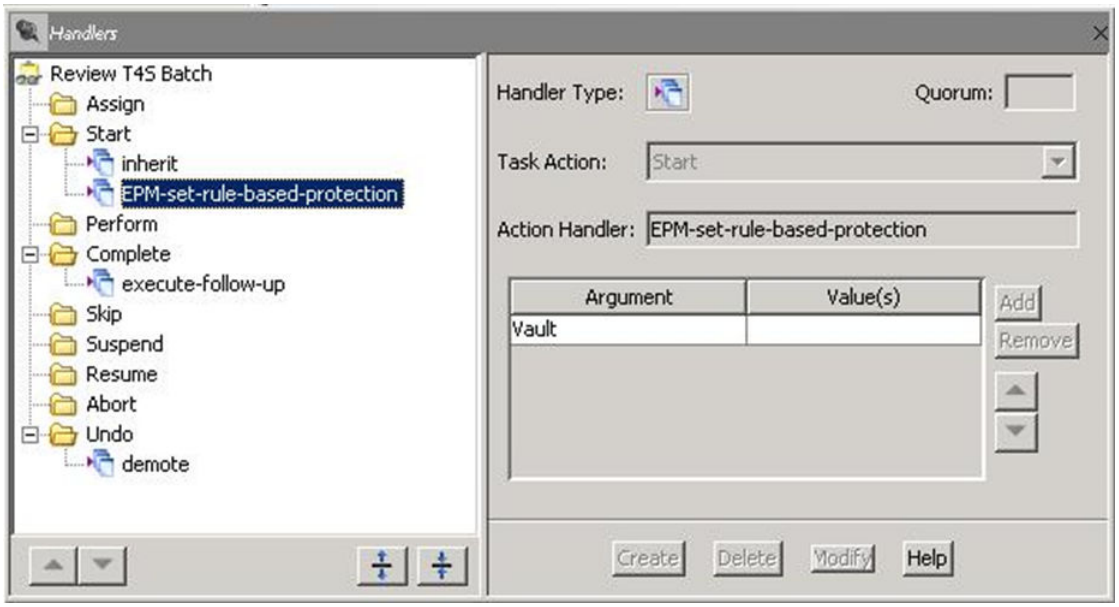
The following screen shots are taken from a T4S installation. For other products some of them may look a bit different, in particular they may differ in notation, e.g. prefix "T4O_", "T4CEP_" or "T4EA_" instead of "T4S_", "EBS" , "CEP" or "EA" instead of "SAP", "Item" instead of "MM" etc.

The following example shows a very simple workflow template setup in order to have the AIG processing done as AIG job instead of processing it in the local Teamcenter client. In fact, it is the standard AIG MM/ Item Workflow Template with only a "Review Task" added:

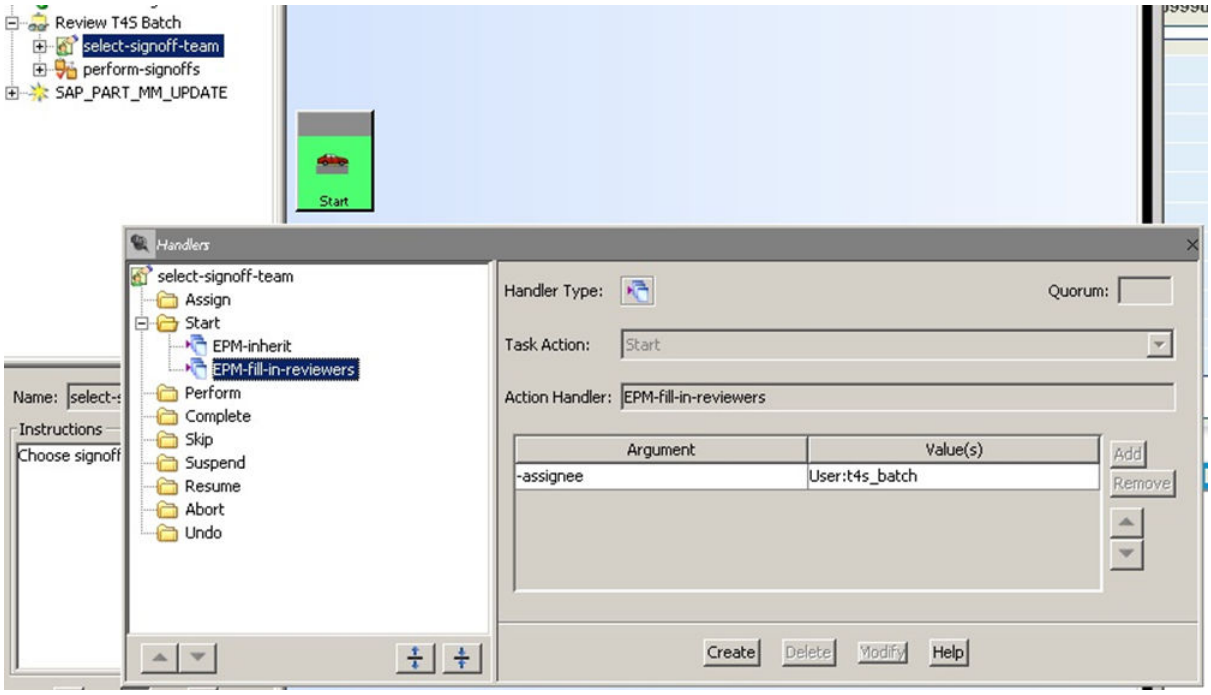


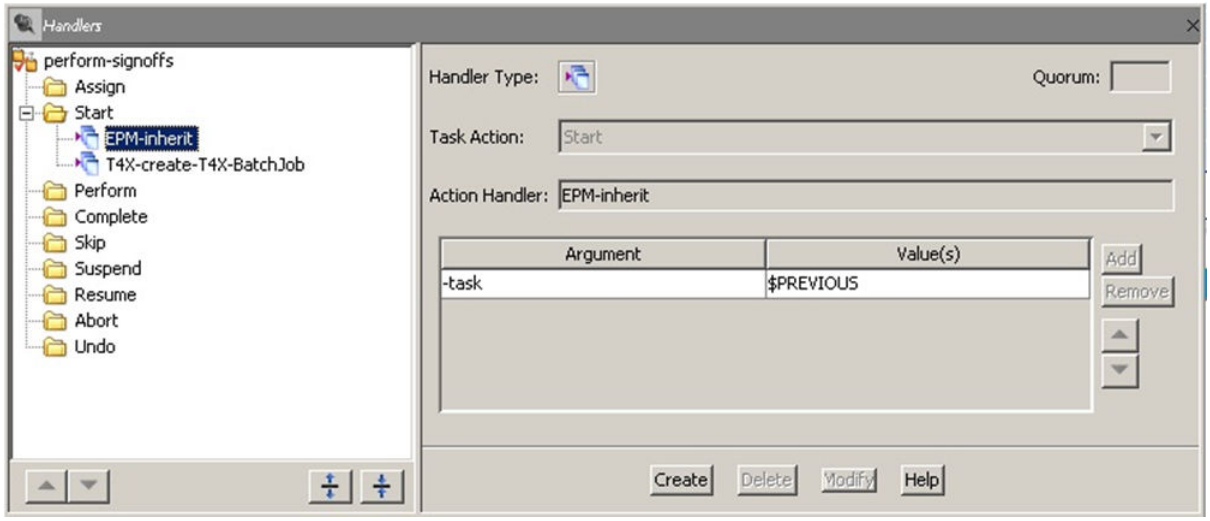
Please note that the "Review T4S Task" (the one that pushes the processing to the Job Server) is located after the "TESIS SAP Logon Task" (T4S), "EBS Logon Task" (T4O) or "EA Logon Task" (T4EA), respectively, so T4x has made sure that there is a correct EA login before the Job Server gets the workflow job (where nobody could type the login credentials).

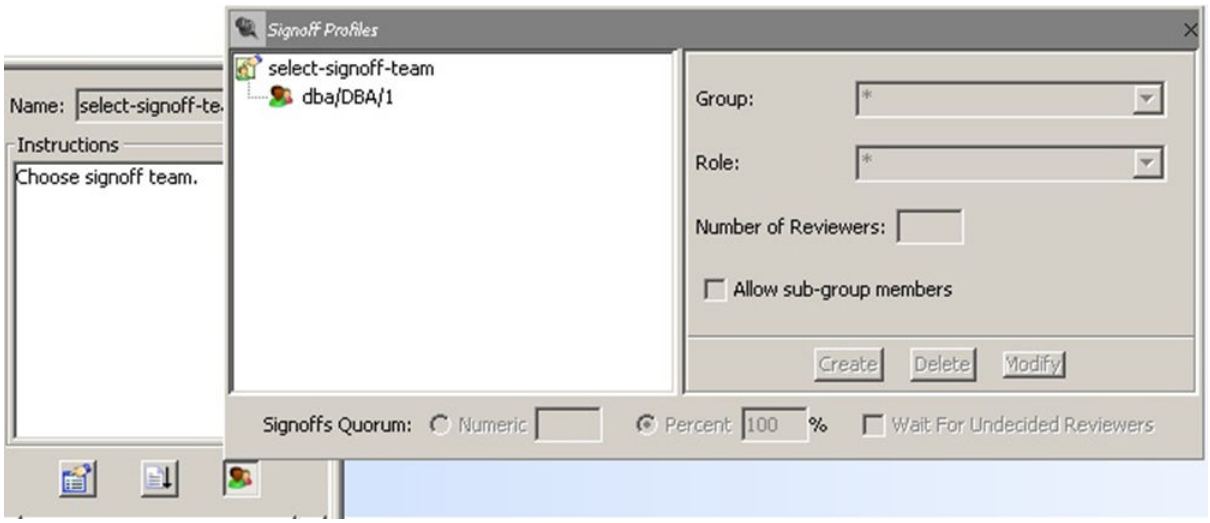
The "Review T4S Task" needs the following settings (of course the argument may be different, according to the desired protection settings):



Please note that the "Review T4S Task" has two sub tasks that need to be adapted as well:

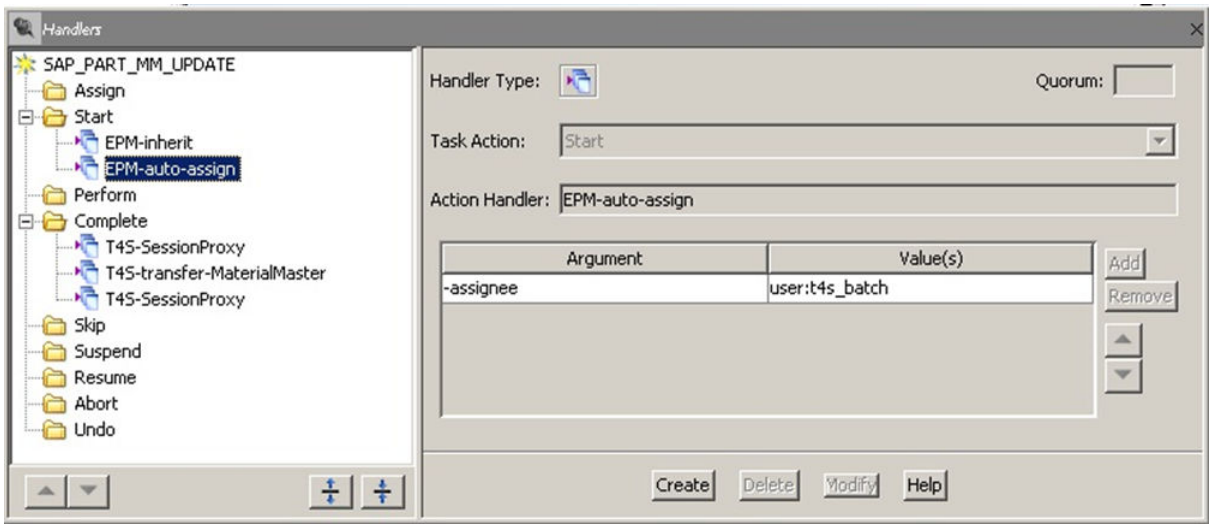






Depending on the Teamcenter configuration (especially regarding Teamcenter user groups and roles rights) and the requirements regarding workflow job processing rights, this setting might need to be different.

The next task should assign the workflow job to the "T4x Batch user":



This setup needs an ERP AutoLogin defined for the mode `BATCH` in *<t4x>_mapping_config.sd*.

As described above, the T4x Job Server will use the EA auto login by default. In order to use the ERP login of the human Teamcenter user who started the workflow, add the following line (or replace the value `AUTOLOGIN` by `WORKFLOW` in mapping file if this line should be present already):

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(ConnectMode4ERP) WORKFLOW
```

In that case, there is no need to specify the ERP AutoLogin in the mapping.

Then modify the workflow template as follows:

After the correct EA login had been checked and before the Job Server gets the workflow job (may be in the same task as the ERP login check), add the action handler

`<T4x>-SessionProxy` with the argument `-Mode=Save`

This will allow the workflow using the same EA login, even if it is processed in another T4x installation (which is the default case with T4x job processing).

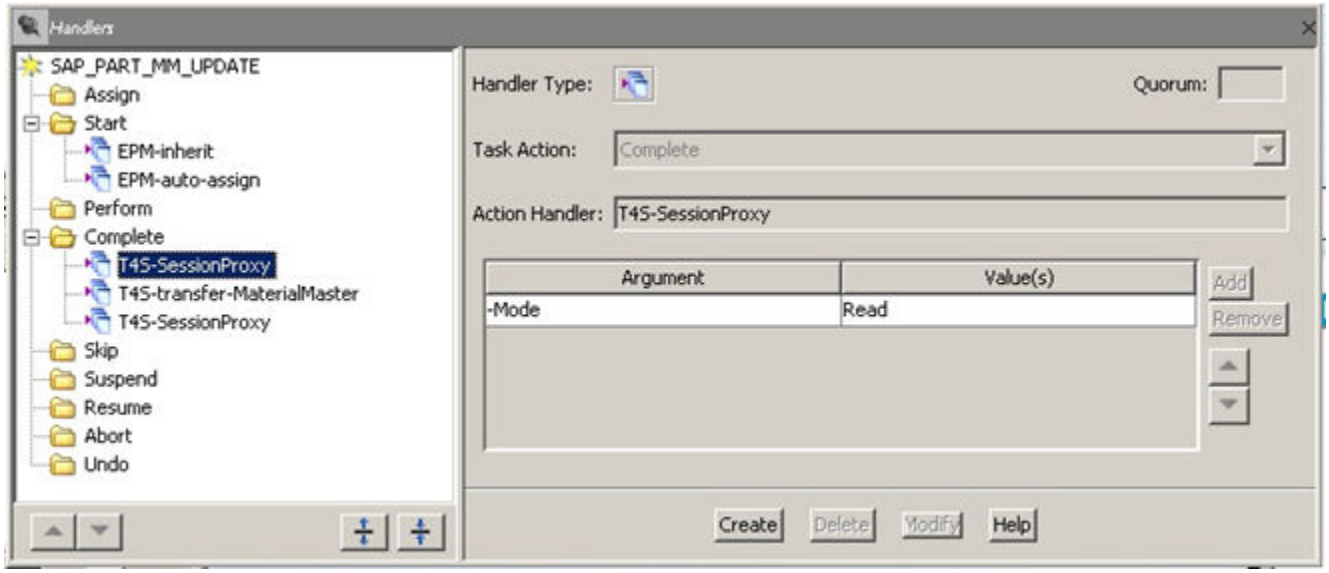
After the Job Server got the workflow job and before the first T4x transfer handler is processed then, add the action handler

`<T4x>-SessionProxy` with the argument `-Mode=Read`

After the last T4x transfer handler, add the action handler

`<T4x>-SessionProxy` with the argument `-Mode=Delete`

Please see the product specific configuration guide for details.



Create Jobs triggered from outside Teamcenter

The typical use case (and only this will be described here) is that there is no action started from within Teamcenter but AIG has to check if there is some data modified in the EA system and save those modifications to Teamcenter. Here are the major steps to trigger updates from the EA system to Teamcenter:

- In some way dependent on your EA system, implement a trigger on an EA internal event. This trigger must be implement using the EA specific configuration and programming language, so it cannot be described in detail here.

- The EA trigger must then be passed to an Active Integration Gateway Gateway Service, e.g. by calling a web service or creating an entry in a database table or a file in a directory watched by Active Integration Gateway. On Active Integration Gateway side, the implementation of this trigger (the web service implementation or the script watching the table or directory) must then initiate an import.

- Prepare the import mapping (e.g. file **<t4x>_mm_item_import_template.sd** in the directory **<T4X_ROOT>/var/mmap/<t4x>**) so that it will fulfill your requirements and source it in the *<t4x>_mapping_config.sd* file. After that, recompile and deploy the mapping.

See also how to **Read and Write Job Attributes**.

5.1.1 Read and Write Job Attributes

T4x allows reading and modifying job attributes from within the mapping. You have to determine the Job ID and then use it read or write the desired data for that specific job.

Determination of the Job ID

From a newly created Job

The function that creates the Job (`::T4X::BATCHJOB::EXPORT::createBatchjob` or `::T4X::BATCHJOB::IMPORT::createBatchjob`) will return the Job ID:

```
# Import Job
set Status [::T4X::BATCHJOB::EXPORT::createBatchjob T4S_TRANSFER_MM\
  T4S_EXPORT_MM $ObjectTag $ObjectClass\
  $ObjectType $ItemId $ItemRevisionId TC2SAP]
set JobID   [lindex $Status 1]

# Export Job
set BatchImport [::T4X::BATCHJOB::IMPORT::createBatchjob T4S_IMPORT_MM \
                $MM_ID TC_IMPORT NONE $ItemData]
set JobID   [lindex $Status 1]
```

Within the mapping of an executed Job

In a case where you want to handle a Job from within a part of the mapping that has been called from that running Job, you can use the following code instead:

```
# This will only work if the mapping runs in the context of a Job.
if {[::API::Core::Batch::isInBatchContext]} {
  set JobID  [::API::Core::Batch::getAttributeFromRawAttributeString \
  ::jobattr JOID]
}
```

> Caution:
>
> The string `::jobattr` is a parameter of the function.

Reading and Writing Job Attributes

Before you can read and write Job Attributes, you need to connect to the BGS Database:

```
::API::Core::Batch::connectTo "BGS"
```

Example on how to read a Job Attribute:

```
set Handle [::API::Core::Batch::connectTo "BGS"]
if {$Handle ne ""} {
  set JobMsg  [::API::Core::Batch::getJobAttribute $JobID message]
  set JobPrio [::API::Core::Batch::getJobAttribute $JobID priority]
}
```

Reading the Job message may be done by this additional function instead:

```
set JobMsg [::API::Core::Batch::getJobMessage $JobID]
```

Example on how to write a Job Attribute:

```
set Handle [::API::Core::Batch::connectTo "BGS"]
if {$Handle ne ""} {
  set rc [::API::Core::Batch::setJobAttribute $BatchJobID priority 15]
  if {$rc ne "1"} {
    return ERROR; # 1=OK, 0=ERROR
  }
}
```

5.2 Configuring Job Processing

The T4x Job Server may be adapted to a large variety of different environments. Therefore it provides a number of settings to modify its behavior to fulfill the different customers' needs as good as possible.

None of the following switches needs to be set explicitly. If not set in the mapping (preferably in the file *<t4x>_mapping_config.sd*), the settings from the following lists are used by default.

Explanation:

- All those switches contain the word "BatchjobDefaults"
- Every job belongs to one of those three groups (see below for details):
  - Workflow: the job was created by a Teamcenter Workflow and can do anything in principle.
  - Import: the job was created by a T4x import script (see **Using Create Jobs on the Job Server**) or a T4x query; it does an "import" to Teamcenter, i.e. in most cases create an object or update it according to data from a EA.
  - Export/Transfer: the job was created by a T4x transfer script (see **Using Create Jobs on the Job Server**); data are exported from Teamcenter to a EA and the other way; this is what every standard T4x transaction started from within Teamcenter does. T4x does not separate "export" (only export data from Teamcenter) from "transfer" jobs (bi-directional between Teamcenter and EA) internally because from the internal technical point of view there is no need to do it: both of them start reading data from Teamcenter, so the names may contain the word "export" but not "transfer"

  All those switches need to be set with respect to the corresponding group (by using their appropriate namespace), i.e. you may define the "same" setting differently for jobs belonging to those groups. Example: define the job retry time (the time that the Job Server will wait after an error occurred before assigning the job to a Job Agent again to process it) to 10 minutes, half an hour and one hour (the value is the time in seconds):
  `set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(DefaultRetryTime) "600"`

```
set ::T4X::BATCHJOB::IMPORT::BatchjobDefaults(DefaultRetryTime:<t4x>_IMPOR
T_<object_type>) "1800"
set ::T4X::BATCHJOB::EXPORT::BatchjobDefaults(DefaultRetryTime:<t4x>_EXPOR
T_<object_type>) "3600"
```

- Most of the switches are for exactly one job type only which is part of the name. This job type name is not correlated at all with the script or workflow name that may have started the job; it is defined internally in T4x depending on the way the job is created and partially the object type(s) it handles, e.g. `<T4x>_EXPORT_<OBJECT_TYPE>`, `<T4x>_IMPORT_<OBJECT_TYPE>`, `T4X_WF_BATCH`.

- The functionality of the switches is rather obvious from their names, e.g. "BatchJobPrio" sets the job priority. It will be explained in the following sections.

Switches for Workflow Based Jobs

Priority

set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(BatchJobPrio) "50"

Timeout

If the job is not processed in the given time, it will be aborted and get the status "Runtime Error"

Number of Retries

Maximum number of attempts to process a job again after an error occurred:

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(MaxRetryCounter) "0"
```

Retry Time

If the `MaxRetryCounter` is not `0` then this controls how long to wait for a retry of the Job.

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(DefaultRetryTime) "3600"
```

Debugging

If set to "ON", additional log entries are created in the job log file

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(T4X_WF_BATCH_DebugSwitch)
"OFF"
```

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(BatchJobTimeout) "3600
```

Auto Login

This tells the Job Server to use the EA auto login only. Set it to "WORKFLOW" if the Job Server should not use the EA auto login but the Teamcenter user has to log in to the EA interactively and this login will be used by this job (needed for the Workflow action handler `<T4x>-SessionProxy`). For T4x this default setting for the EA Login is not supported at the moment.

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(ConnectMode4ERP)
"AUTOLOGIN"
```

Rescan Time of the inbox

After the WF Job has been created, the workflow stops and there will be a Task in the inbox of the assigned Job user. This is the time in seconds after that the Job Server will check the Inbox again in order to find missing jobs:

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(InboxRescanTimeSecs4MissingJ
obs) "60"
```

Shut down ITK pipe

Shut down the ITK pipe in case of an error. Takes slightly more time (maybe considerable when many jobs are to be processed) but prevents errors resulting from inactive ITK pipes.

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(StopITK) "TRUE"
```

Trigger Comment

Adds a comment to the approved task of the worklfow.

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(TriggerComment) "Triggered
by T4x Job Server"
```

Task Status Validation

It is possible to activate a build-in workflow TaskStatus validation (that includes the successor task status if possible) to detect an application error or retry situation.

```
set ::T4X::WORKFLOW::BATCHJOB::BatchjobDefaults(useTaskStatusValidation)
true
```

Import Jobs

Switches for import jobs (triggered from outside Teamcenter):

Priority

```
set ::T4X::BATCHJOB::IMPORT::BatchjobDefaults(BatchJobPrio:<job_type>) "20"
```

Timeout

```
set ::T4X::BATCHJOB::IMPORT::BatchjobDefaults(BatchJobTimeout:<job_type>)
"3600"
```

Number of Retries

```
set ::T4X::BATCHJOB::IMPORT::BatchjobDefaults(MaxRetryCounter:<job_type>)
"1"
```

Retry Time

```
set ::T4X::BATCHJOB::IMPORT::BatchjobDefaults(DefaultRetryTime:<job_type>)
"3600"
```

> Caution:
>
> Lower values for `BatchJobPrio` result in a higher priority.
>
> The job priority works in conjunction with the job pattern. For details about job pattern see chapter **AIG Job Agent Configuration** in the **Active Integration - Installation Guide**

5.2.1 Managing Job Dependencies

In some cases it is not enough to manage the order of processed jobs by priority. Job Dependencies allow to control the start behavior of one Job (Head Job) that is depending on the result of another (Tail Job). Job dependencies can only be set when the Head Job is created.

Define Hard Dependencies

A dependency is satisfied if and only if all Tail Jobs are in finished state.

The following function can be used to define a hard dependency together with `::T4X::BATCHJOB::EXPORT::createJob`:

```
::T4X::BATCHJOB::IMPORT::setHardDependencyList
::T4X::BATCHJOB::EXPORT::setHardDependencyList
```

Define Soft Dependencies

A dependency is satisfied if and only if all Tail Jobs are not in state ready or running any more. So they have finished with an application or runtime error or successfully.

Define a list of Job Dependencies as input of the

```
::T4X::BATCHJOB::EXPORT::setSoftDependencyList
```

The following function can be used to define a Hard Dependency together with `::T4X::BATCHJOB::EXPORT::createJob`:

```
::T4X::BATCHJOB::IMPORT::setSoftDependencyList
::T4X::BATCHJOB::EXPORT::setSoftDependencyList
```

5.2.2 Configuring Time Windows

The Time Window of a Job defines during which time frame a Job in the pool can be processed. By default there is no restriction.

Via Workflow Handler

The handler `T4X-create-T4X-BatchJob` has the optional parameters `-TimeWindowStart` and `-TimeWindowEnd`.

Via ITK Function

The window can be configured in the input dictionary for the Job creation. Please see the documentation of the procedures `T4X::BATCHJOB::IMPORT::createJob` and `T4X::BATCHJOB::EXPORT::createJob`. Here is an example:

```
set JobAttributeDict [::T4X::BATCHJOB::EXPORT::setJobAttributes \
$JobAttributeDict JOB_TIME_WINDOW_START 20:00]
set JobAttributeDict [::T4X::BATCHJOB::EXPORT::setJobAttributes \
$JobAttributeDict JOB_TIME_WINDOW_END 00:00]
```

5.3 Job Alerts

The Job Server alerts are an observation and notification mechanism regarding the Job Server status. The Job Server status is observed on a regular basis and notification e-mails are sent, e.g. when an error occurs.

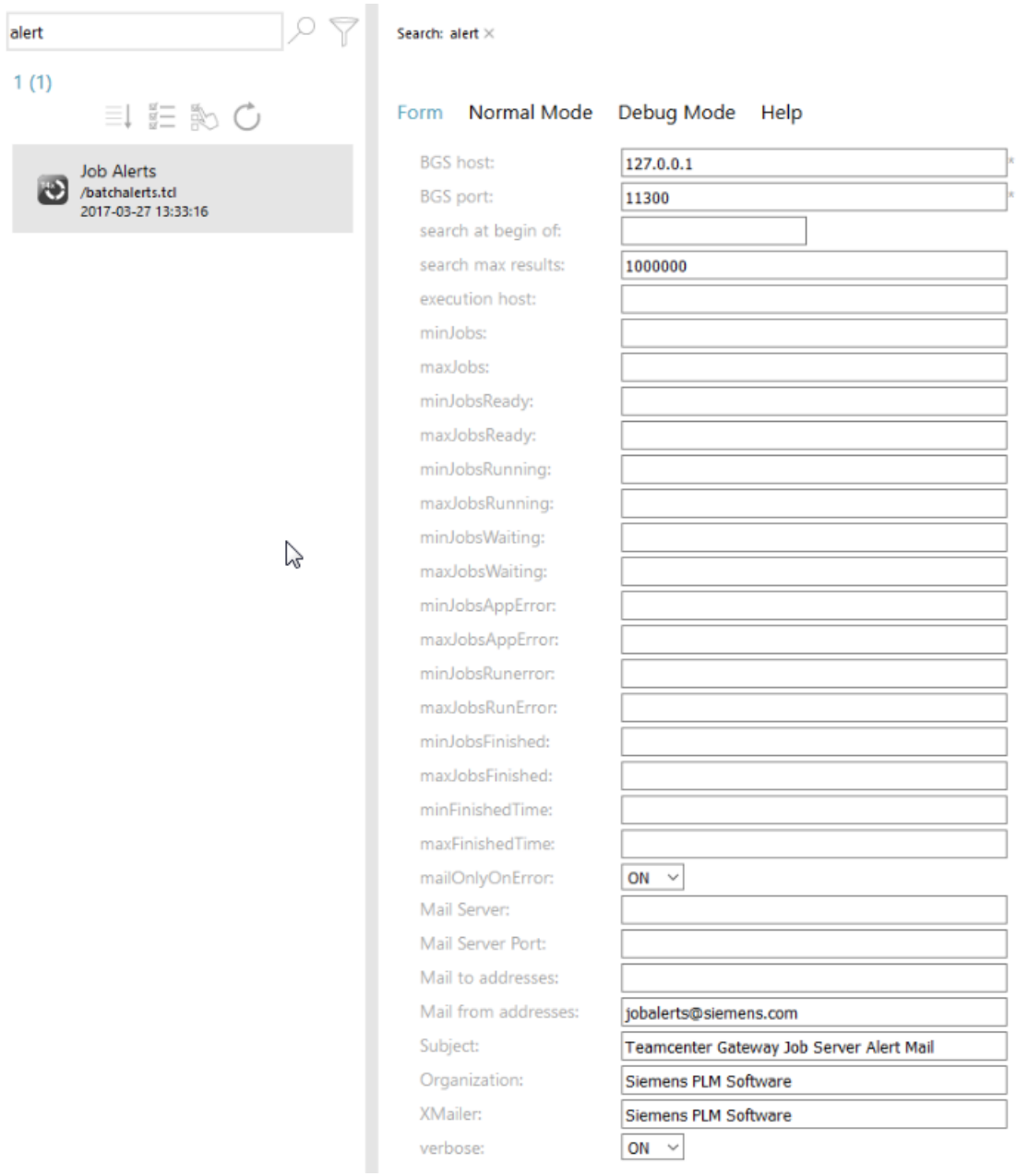
Examples of Job Server error scenarios are the following:

- too many jobs with an error status
- too long execution time of jobs

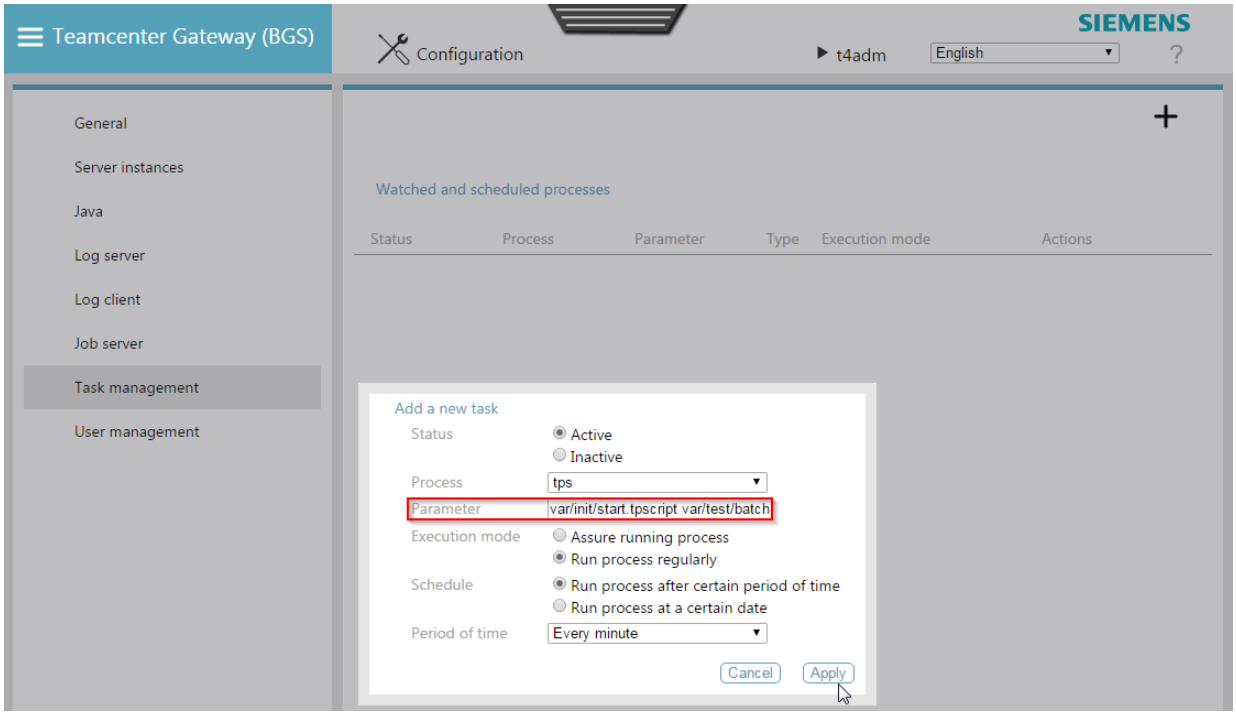
The Job Server Alerts are implemented as a test script and are available for BGS only.

The alert script can be executed in two ways:

- manually (run the script in the Admin UI "script environment" application)



- automatically (run the script by a scheduled task in the Admin UI configuring it in the menu entry Configuration – Task Management
using parameters, see below)



Example for the parameters (see graphic above):

```
var/init/start.tpscript var/test/batchalerts.tcl
 -ServerIp 127.0.0.1 -ServerPort 11300
 -searchTimeStart {} -searchMaxResults 1000000
 -executionHost {} -minJobs 112222 -maxJobs {}
 -minJobsReady {} -maxJobsReady {}
 -minJobsRunning {} -maxJobsRunning {}
 -minJobsWaiting {} -maxJobsWaiting {}
 -minJobsAppError {} -maxJobsAppError {}
 -minJobsRunError {} -maxJobsRunError {}
 -minJobsFinished {} -maxJobsFinished {}
 -minFinishedTime {} -maxFinishedTime {}
 -mailServerIp {} -mailServerPort {} -mailToAddresses {}
 -mailFromAddress {}
 -mailSubject {Teamcenter Gateway Job Server Alert Mail}
 -mailOrganization {Siemens PLM Software}
 -mailXMailer {Siemens PLM Software}
 -mailOnlyOnError OFF -v ON +language en
```

The alert script performs the following steps:

1. Job search: all jobs to consider are retrieved. By default, those are all jobs created up to an hour ago and at most 100000 jobs. The exact settings used for the job search depend on the "search parameters".

2. Job analysis: all retrieved jobs are examined. The jobs are classified depending on their job status. The number of jobs with status "ready", "waiting", "finished", ... is determined. For jobs with status "finished" also the execution time is considered. The minimum execution time and the maximum execution time of jobs having status "finished" are determined. An error scenario is recognized, if some job number value exceeds some defined cardinality limit or if some job execution time value exceeds a defined time limit. The exact settings used for the job analysis depend on the "counter parameters" and on the "timer parameters".

3. Job notification: a notification e-mail is sent, if one or more error scenarios have been recognized. It is also possible to specify that notification e-mails have to be sent always. The exact settings used for the job notification depend on the "mail parameters".

The alert script has external parameters. Those parameters are arguments in case of scheduled runner or values in text boxes in case if you use the Admin UI to trigger the script manually.

In the following sections the configuration parameters are explained.

Mail parameters

Mail parameters specify the settings of the e-mail server used for e-mail notification and whether notification e-mails have to be sent only on error or always.

The following principal mail parameters exist:

Parameter/Input field	Description	Default value
`mailOnlyOnError`	specifies when a notification e-mail is sent (0 = send an e-mail every time the script is executed, 1 = send an e-mail if an error occurred)	ON
`mailServerIp`	IP address of the e-mail server to be used for notification e-mails	
`mailServerPort`	port number of the e-mail server to be used for notification e-mails, for example 25	0
`mailToAddresses`	target e-mail addresses to be used for notification emails	
`mailFromAddress`	Source e-mail address	jobalerts@siemens.com
`mailSubject`	e-mail subject	Teamcenter Gateway Job Server Alert Mail
`mailOrganization`	e-mail organization	Siemens PLM Software
`mailXMailer`	e-mail x-mailer	Siemens PLM Software

Search parameters

Search parameters specify the settings used for retrieving jobs. By default, the search parameters specify that all jobs created up to 1 hour ago and at most 100000 jobs are retrieved.

The following search parameters exist:

Parameter/Input fields	Description	Default value
`searchTimeStart/ search at begin at`	Specifies the datetime job for creation time. All jobs which created after this value are considered.	
`searchMaxResults/ search max results`	Maximum number of jobs considered	100000

> Caution:
>
> Normally it is not necessary to change the search parameters.

Counter parameters

Counter parameters specify settings used for the job analysis. A counter parameter is either a cardinality lower limit or a cardinality upper limit. The retrieved jobs are compared against those limits. If a limit violation subsists an error scenario is recognized. By default, the counter parameters specify disabled cardinality limits set to "". To enable a parameter it has to be set to the value to be checked.

The following counter parameters exist:

Parameter	Description	Default value
`minJobs`	minimum number of jobs allowed	
`maxJobs`	maximum number of jobs allowed	
`minJobsReadyr`	minimum allowed number of jobs having the job status ready	
`maxJobsReady`	maximum allowed number of jobs having the job status ready	
`minJobsRunning`	minimum allowed number of jobs having the job status running	
`maxJobsRunning`	maximum allowed number of jobs having the job status running	
`minJobsWaiting`	minimum allowed number of jobs having the job status waiting	
`maxJobsWaiting`	maximum allowed number of jobs having the job status waiting	

Parameter	Description	Default value
minJobsAppErr	minimum allowed number of jobs having the job status application error	
maxJobsAppErr	maximum allowed number of jobs having the job status application error	0
minJobsRunErr	minimum allowed number of jobs having the job status runtime error	
maxJobsRunErr	maximum allowed number of jobs having the job status runtime error	0
minJobsFinished	minimum allowed number of jobs having the job status finished	
maxJobsFinished	maximum allowed number of jobs having the job status finished	

Timer parameters

Timer parameters specify settings used for batch job analysis. A timer parameter is either a timer lower limit or a timer upper limit. The retrieved jobs having state "finished" are compared against those limits. If a limit violation subsists an error scenario is recognized. By default, the timer parameters specify disabled timer limits set to "". To enable a parameter it has to be set to the value to be checked.

The following timer parameters exist:

Parameter	Description	Default value
minFinishedTime	Minimum execution time allowed for jobs in "finished" state	
maxFinishedTime	Maximum execution time allowed for jobs in "finished" state	

5.4 Job Server Hints and Troubleshooting

- **Cancel a Job and Return SKIPPED**.
- **Job Pool Functions Script on Job Server**.
- **Store, Retrieve or Delete Persistent Data on the BGS**.

5.4.1 Cancel a Job and Return SKIPPED

You may configure a job to cancel and return the value "SKIPPED". Then this job is finished and the ERP Z-table (if configured) will store the value C (= Cancelled). This may be done in the following mapping (`getObjectInfo`) functions:

For T4S as example:

- `::T4S::BOM::CUSTOM::ITEM::IMPORT::getImportBomInfo`
- `::T4S::MM::CUSTOM::ITEM::IMPORT::getMaterialMasterInfo`
- `::T4S::ECM::CUSTOM::ITEM::IMPORT::getChangeMasterInfo`
- `::T4S::DIR::CUSTOM::ITEM::IMPORT::getDocumentInfoRecordInfo`
- `::T4S::PO::CUSTOM::CC::IMPORT::readSAP_ProductionOrder`
- `::T4S::POBOM::CUSTOM::ITEM::IMPORT::getImportPoBomInfo`

5.4.2 Job Pool Functions Script on Job Server

The BGS Admin UI script "Job Pool Functions" provides several functions – import, export and deletion - for maintaining the complete Job Pool. You can use this script to create a "dump" of your Job Pool (export), delete your existing pool and restore your previously exported jobs on the same or even another Job Pool (import).

The script provides four different functionalities described in detail beneath (select from the "Function" dropdown). For each function you need to enter a user name and password to execute it. Depending on the chosen function, different optional and mandatory parameters need to be provided.

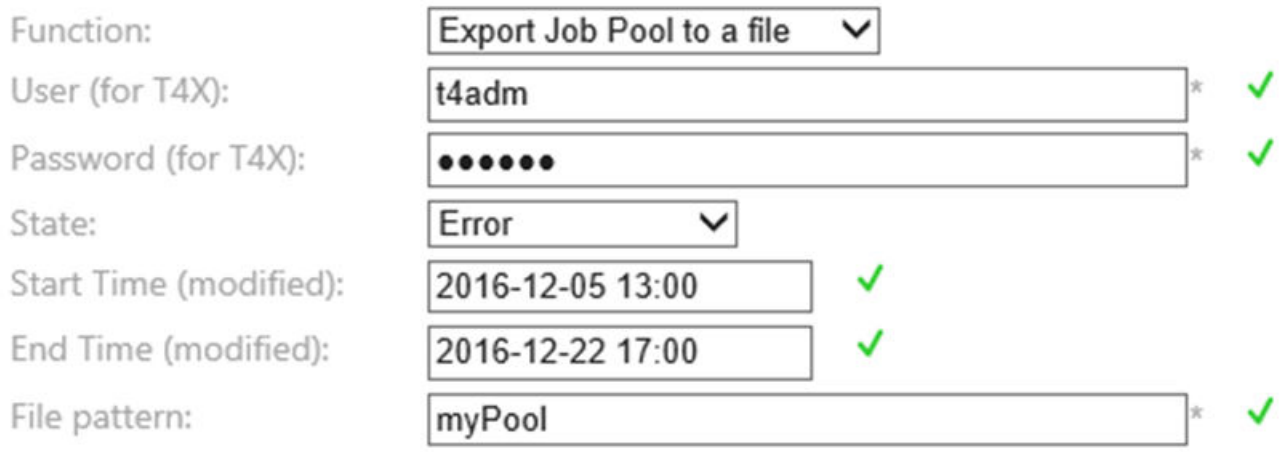
Export Job Pool to a file

Use this function to dump (export) jobs matching certain criteria to one or more files located in *<BGS_ROOT>/tmp*. You need to specify a name for the exported file(s) ("File pattern"). The files will be called *<filename>_<number>.cbu.00*, *<filename>_<number>.cbu.01* and so on, depending on the number of jobs exported.

Optionally, you can restrict the number of jobs to be exported based on their state or modification date by providing the according parameters.

For example, if you choose "Error" in the "State" filter, only jobs that are in the state "Application Error" or "Runtime Error" are exported. Similarly, you can define a "Start Time (modified)" and/or "End Time (modified)" to export only jobs having a "Modified" time stamp in the given time frame. If you define a "Start Time", but no "End Time" all jobs modified since the start time up to now are exported. Otherwise

if you define an "End Time", but no "Start Time" all jobs modified until the defined timestamp are exported.



Reset Job Pool

This function deletes all jobs in the pool matching certain criteria. Similar to the export function you need to provide user name and password and can further specify which jobs to delete by defining their "State" or their modification date ("Start Time" and "End Time"). For more details on these filters, please see the previous section.

This does not delete any files! If you want to delete jobs due to a full hard disk, then please execute this function, stop your BGS and physically delete all files in *<BGS_ROOT>/var/pool* and *<BGS_ROOT>/var/log/bajob*

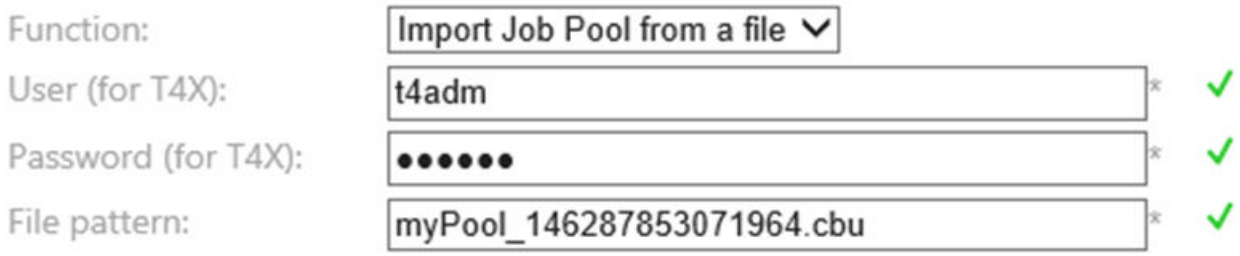
Import Job Pool from a file

Use this function to import all jobs from a previously created export file. You need to copy the exported files to *<BGS_ROOT>/tmp* and provide the correct name as "File pattern".

For example, if you have two files

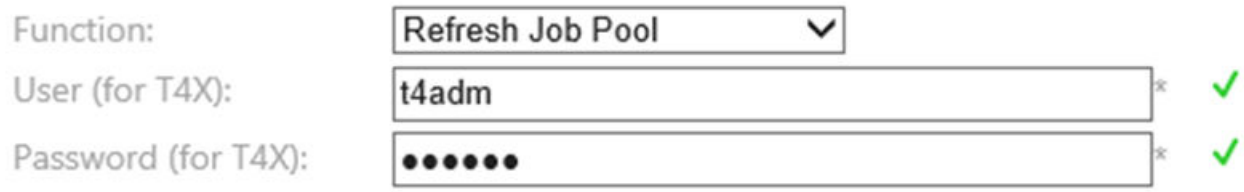
- **myPool_146287853071964.cbu.00**
- **myPool_146287853071964.cbu.01**

you need to copy both into the *tmp* folder and write **myPool_146287853071964.cbu** into the "File pattern" textbox to import the whole pool again.



Refresh Job Pool

Use this functionality to export, clean and import your whole Job Pool at once.



5.4.3 Store, Retrieve or Delete Persistent Data on the BGS

A normal use case to store data persistently is to save it in an attribute of an object in Teamcenter, a connected system or in a file on the disk. However, another option is to store it on the BGS. This can be useful if the information is needed only for a defined period of time and can be forgotten after this time period ends. In addition to that, the information can be accessed by all GS that are connected. Example code:

```
# required data
set key     "MyProdFlagIdentifier"
set value   "MyProdFlagValue"
set timeout 3600; # set to 3600 seconds
set storage "share"; # keep data if BGS restarts
# set storage "local"; # delete data if BGS restarts
```

```
# set flag
::PL4X::SHM::remote_shmset BGS $storage $key $timeout $value

# read flag
set readValue [::PL4X::SHM::remote_shmget BGS $storage $key]
puts "The value of my flag is $readValue"

# delete flag
::PL4X::SHM::remote_shmdelete BGS $storage $key
```

There are also some additional functions:

```
::PL4X::SHM::remote_shmexists
::PL4X::SHM::remote_shmgettout
::PL4X::SHM::remote_shmsettout
```

Caution:

- This functionality is not intended for replacing a database to store a huge amount of data over a longer period of time.
- Stored data cannot be easily migrated or copied to another BGS.
- Any values stored in the local storage will be lost when the BGS is restarted.
- The data should be deleted or an appropriate timeout should be set to avoid that the BGS consumes too much memory.

6. T4x Logging

6.1 Configure Security Context on Log Channels and Log Lines

Security context level

Log channels and log lines can be assigned to a security context level.

Currently the following security contexts are defined:

- UNRESTRICTED
- RESTRICTED
- CONFIDENTIAL
- STRICTLYCONFIDENTIAL

Security level restricts the access to the log channels and log lines for certain user roles. Security context could be also assigned to single log line. Note that the default security context of all log channels is UNRESTRICTED.

AIG user roles

The following table describes how the AIG user roles are mapped to the security contexts:

User role	Security context
User	UNRESTRICTED
Support	RESTRICTED
Operator	CONFIDENTIAL
Administrator	STRICTLYCONFIDENTIAL

Set security context on a log channel

In a session log:

```
::PL4X::CORE::setSecurityContext4SessionLogChannel RESTRICTED
```

In a transaction log:

```
::T4X::TRANSLOG::setSecurityContext4TransactionLog $TransactionId
RESTRICTED
```

In any other logs

```
# create logchannel index first
set LogChannelPath "/tmp/Custom.log"
::LogTools::createLogChannel $LogChannelPath
# then change security context
::LogTools::setSecurityContext4LogChannel $LogChannelPath RESTRICTED
# and finally write message
tpwrite -logchannel $LogChannelPath -mtype INTERN "+++++  test info ++++
+"
# increase security context
::LogTools::setSecurityContext4LogChannel $LogChannelPath CONFIDENTIAL
```

Set security context on a single log line

In a session log:

```
tpwrite -logchannel [::PL4X::CORE::getSessionLogChannel]\
 -mtype NOTE -sctx CONFIDENTIAL "+++++ confidential info +++++"
```

In a transaction log:

```
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId INFO\
   "+++++ confidential info +++++" CONFIDENTIAL
```

or

```
::T4X::TRANSLOG::writeCustomReverseMappingLog $TransactionId INFO\
   "+++++ confidential info +++++" CONFIDENTIAL
```

In any other logs

```
tpwrite -logchannel "tmp/Custom.log" -mtype NOTE\
  -sctx CONFIDENTIAL "+++++ confidential info +++++"
```

Set security context on an attachement

```
# write a log line with STRICTLYCONFIDENTIAL attachment
set xml "<testtag>hello world</testtag>"
::LogTools::writeLogMessageWithAttachment
[::PL4X::CORE::getSessionLogChannel]\
  "+++++  test info with link to +++++" "My XML file" $xml "text/xml"
INTERN\
  "XML-TEST.xml" STRICTLYCONFIDENTIAL
```

Set security context on a log line and on an attachement

```
# write a STRICTLYCONFIDENTIAL log line with STRICTLYCONFIDENTIAL
attachment
set xml "<testtag>hello world</testtag>"
::LogTools::writeLogMessageWithAttachment
[::PL4X::CORE::getSessionLogChannel]\
  "+++++  test info with link to +++++" "My XML file" $xml "text/xml"
INTERN
  "XML-TEST.xml" STRICTLYCONFIDENTIAL STRICTLYCONFIDENTIAL
```

Teamcenter object driven security context

In the context of a Teamcenter based workflow transfers it is also possible to specify by preferences and BMIDE rules an object driven security context for the transaction log channel. The following preferences are possible:

- <ProductKey>_<TargetType>_TransactionSecurityContext_STRICTLYCONFIDENTIAL_<TcObjectType>
- <ProductKey>_TransactionSecurityContext_STRICTLYCONFIDENTIAL_<TcObjectType>
- T4X_TransactionSecurityContext_STRICTLYCONFIDENTIAL_<TcObjectType>
- <ProductKey>_<TargetType>_TransactionSecurityContext_CONFIDENTIAL_<TcObjectType>
- <ProductKey>_TransactionSecurityContext_CONFIDENTIAL_<TcObjectType>
- T4X_TransactionSecurityContext_CONFIDENTIAL_<TcObjectType>
- <ProductKey>_<TargetType>_TransactionSecurityContext_RESTRICTED_<TcObjectType>
- <ProductKey>_TransactionSecurityContext_RESTRICTED_<TcObjectType>
- T4X_TransactionSecurityContext_RESTRICTED_<TcObjectType>
- <ProductKey>_<TargetType>_TransactionSecurityContext_UNRESTRICTED_<TcObjectType>
- <ProductKey>_TransactionSecurityContext_UNRESTRICTED_<TcObjectType>
- T4X_TransactionSecurityContext_UNRESTRICTED_<TcObjectType>

If no security context could be assigned during the check of the sequence described above, the system will use the UNRESTRICTED security context.

The following example shows how to define the security context:

<T4x>_<TargetTypeName>_TransactionSecurityContext_STRICTLYCONFIDENTIAL_<EA>2_<T4x>_Item Revision= isTrue

In this case all <EA>2_<T4x>_Item Revision type based transaction log channels use the STRICTLYCONFIDENTIAL security context type, but only for the <T4x>_<TargetTypeName> transfer.

T4X_TransactionSecurityContext_CONFIDENTIAL_ItemRevision= isTrue

In this case all ItemRevision type based transaction log channels uses the CONFIDENTIAL security context type.

6.2 Create T4x error messages

T4x allows creating of user defined error messages at any reasonable point in any of the mapping files. The following example shows the code for an error message in the BOM mapping to indicate that there is no EA Material number stored for a Position in the Teamcenter BOM, so the corresponding Material will not be found in EA:

```
if {$MaterialNumber eq ""} {
  ::T4X::CORE::storeMessage2 "BOM Mapping" "BOM error: no EA Material
number
    stored for Pos $Position. Does the Material already exist?" ERROR
  return Error
}
```

Please note that:

- The line `return Error` causes T4x to abort the processing of this BOM completely. If you only want to skip the current position, `return SKIPPED` instead. Then the BOM will be created or updated in the EA, but without that position. A premature `return OK` would try the transaction with these incomplete data as far as they have been set up to this `return`; therefore the transaction will fail in most cases.

- If a message was defined with `storeMessage2` and the transaction aborted with `return Error` this message will be shown to the Teamcenter user:

  - In a Workflow, the message will be shown in Teamcenter's Workflow popup window.

  - If the Workflow log file Dataset attachment is used (by the rule handler T4X-attach-LogfileDataset), this message will be written there, too.

  - The message will be written to the T4x transaction log and the T4x session log as well.

- The function signature is as follows:

```
proc storeMessage2 {MessageSource Message {Type INFO} {Format NOHEX}}.
```

- So you may skip the value `ERROR` in your call (same as setting `INFO` instead), then the message will be shown as info in the logs, not as an error (which is marked in red).

- Set the Type value to `WARNING` in order to show the message in orange (in the T4x session log, it will be marked with `T4X-WARNING`).

- The parameter `MessageSource` allows specifying where the message comes from, e.g. it was created from within the "`BOM Mapping`" as shown in the example above.

- The last parameter (`Format`) determines if the message is stated as plain text or in hex format, in most cases you may leave it, so T4x will treat it as plain text.

6.3 Save a session log file info in a way to find it easily

As T4x writes a new session log file whenever Teamcenter is started there may be many T4x session log files if Teamcenter is started often. In order to later find easier the T4x session log file where a specific process had been done, a specific data can be written there in order to find the correct log file later easily. The following example shows it with the string MyLogKey12345. That can be set for example for checking it with T4x Material actions (because that are the most frequently used ones), in the Material mapping function (e.g. in T4S: `TC_Object2SAP_MaterialMaster`). So after that function is ready, no matter if the same Teamcenter session is still running or not, that T4x session log can be found easily in the BGS Admin UI menu "Session log files" (click "Session" under the main menu point "Log files") by just adding the word MyLogKey12345 into the search line:

```
set rc [tpco_setLogIndex -logchannel [::T4X::CORE::getSessionLogChannel]
\
  -key abc -value MyLogKey12345]
```

The parameter `-key` is needed internally but from user point of view it does nothing, so it has the simple value "abc" here only. The return code (stored as variable `rc`) should be "0" (zero), else there was an error.

Another example for creating a value with the current date and machine name in order to let the users search their T4x session log files more easily if there are many users working with the same BGS which means that searching the current session logs shows many instead of the own one only. So using the following section in any mapping source can help you searching your session log easier with the name of your computer, no matter how many colleagues create any session log files:

```
set MyLogValue  "$::env(COMPUTERNAME)_[clock format [clock seconds] \
 -format "%d_%H"]"
set rc [tpco_setLogIndex -logchannel [::T4X::CORE::getSessionLogChannel]
\
  -key abc -value $MyLogValue]
# Maybe write that name into the transaction log to find it easily:
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId NOTE
 "+++ stored the name '$MyLogValue' in the T4S session log returns '$rc'"
```

In order to find the created string easier, that example only uses the values of the current day and hour, not the whole date and time which would be more complicated. So just ask your OS in a command shell how your computer is named internally (in Windows: `set COMPUTERNAME`, in Linux and UNIX just the `hostname`) and notice the time when you started Teamcenter and T4x.

Example: the used computer has the name "Comp123" and Teamcenter was started between 8 AM and 9 AM on January 13, then that name will be "Comp123_13_08" which can be searched directly in the T4x Admin UI "Session log files".

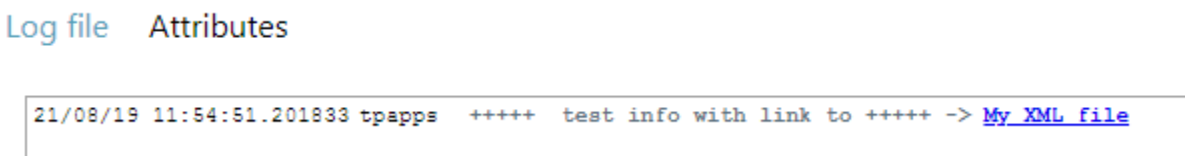
> Caution:
>
> Depending on internally settings, that hour number can be "8" or "08", so maybe actually check that name result in the additional transaction log output.

6.4 Add a file link to a T4x log file

In order to not just write text information into a log file but create a whole new file and storing a link on that file into any of the T4x log files, use a function call like the following example:

```
set xml    { <testx>   <aaa>Test XML file content</aaa>   </testx> }
::LogTools::writeLogMessageWithAttachment "tmp/myCustom.log" "+++++
test info with link to +++++" "My XML file" $xml "text/xml"
```

- The file is sent to your configured BGS.
- This will create a new T4x log file *tmp/myCustom.log*.
- Please remember to not try and write and own log file outside the log *tmp* directory.
- The parameter `text/xml` string defines how the web browser will handle that link; in this case as xml file, but other types are possible as well.
- When clicking the file tmp/myCustom.log in the BGS Admin UI log application, it will show the following content:

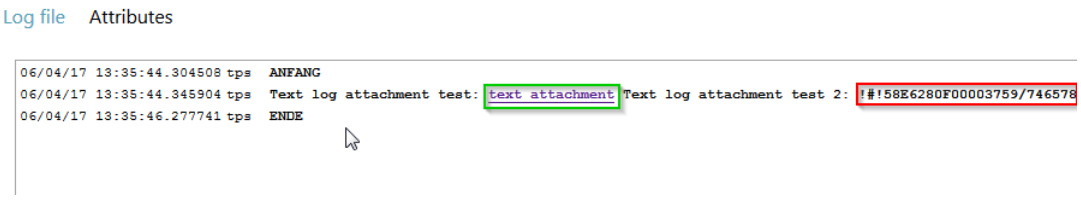


- Click the link (the underlined part) and the web browser should show you the defined text "Test XML file content" in some XML formatting, maybe with an additional message that it does not find a corresponding XML style sheet to display it correctly.

- You may create such a link in every T4x BGS log file but not in additional files without HTML formatting. So especially it cannot work when writing to the Workflow attachment log file which is pure text only.

Caution:

T4x can not handle more than one log attachment per log line:



6.5 Create T4x error messages in the current Teamcenter language

The chapter above describes how to define T4x error messages that are stated in the mapping as fix texts. To show these messages in the language you are currently using in Teamcenter, do the following:

- Adapt Teamcenter's own language specific error messages file, e.g. *%TC_ROOT%\lang\textserver \en_US\ue_errors.xml*.

- In this file, define your messages with an error number, e.g.:
  `<error id="4">User error 4: Item %1$ has problem '%2$'</error>`.

- Note that the "error base" is set fix to 919000 in this file. This means, if you are using Teamcenter in English and an error with the number 919004 occurs (no matter if Teamcenter itself or any plug-in triggered it), then Teamcenter will show the error message "User error 4: ..." and the `%1$` is replaced by an additional text created by the initiator of that error.

- Define your error message in the mapping file with the following function:

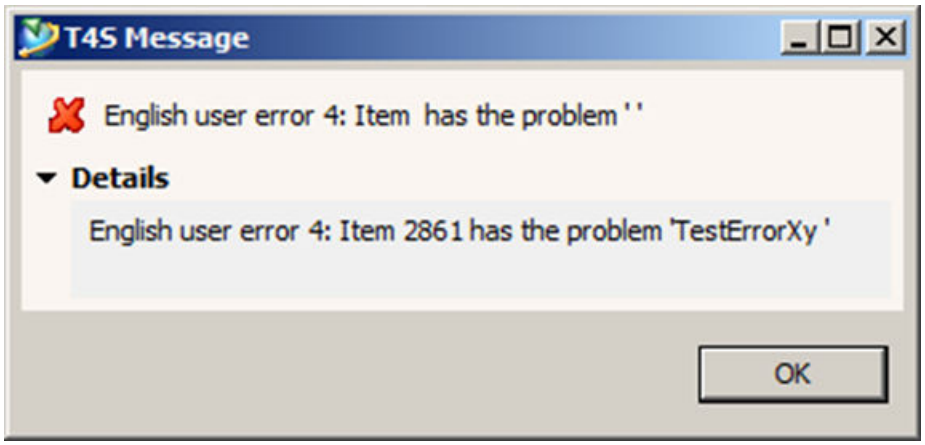
  ```
  ::T4X::CORE::storeUserMessage ERROR <ErrorNo> <S1> <S2> <S3> <S4> <S5>
  ```

  `<ErrorNo>` is the complete error number, i.e. not "4" in this example, but "919004". The parameters `S1` to `S5` are optional (this function supports up to five); if specified, their values replace the `S1` to `S5` in the error message definition in *ue_errors.xml*.
  Example:

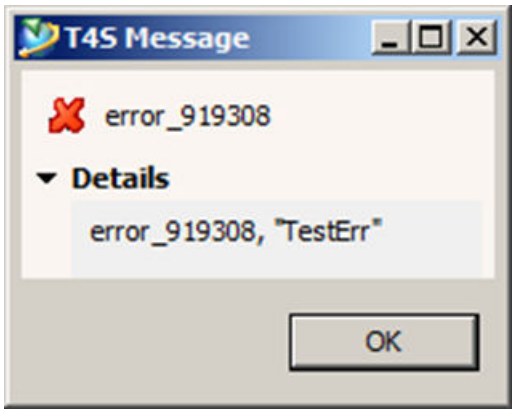
  ```
  ::T4X::CORE::storeUserMessage ERROR 919004 $ItemID TestErrorXy
  return ERROR   ;# only if the function returns ERROR, T4x shows such an
                    error message and stops the transaction at that point
  ```

Note that:

- If the T4x mapping function which contains that additional function returns `OK`, then no error message is shown, no matter how it is defined. If it returns `ERROR` then the transaction ends with error and that defined error message is shown to the Teamcenter user in the same way as any other error, i.e. as a popup window or in the Workflow window (a Workflow job is stopped then). Example with the above shown settings in a T4S portal transaction:



- If the error number stated in that mapping call `storeUserMessage` is not defined in the Teamcenter internal error messages file ue_errors.xml, then T4x shows exactly the message as defined in that call, e.g.:
`::T4X::CORE::storeUserMessage ERROR 919308 TestErr`



6.6 Write Custom Log Messages

Use the command `tpwrite` to write your own additional information into a log file from within the mapping. Example:

```
tpwrite "My test output"
```

This will write the given string "`My test output`" to the tpapps log file which is located in the logs under *sys/<machine_name>/<instance_name>/tpapps64.log*. The command writes into only one log.

The list of optional parameters for `tpwrite` is:

- `-logchannel <log_file>`
- `-mtype DEBUG|ERROR|WARNING|INTERN|NOTE`
- `-stacklevel <level_number>`

The Parameter **-logchannel** defines the name of the log file (see Session- and User Log below).

In order to write your log message into a log in a specific color, use the parameter **-mtype** (message type):

```
tpwrite -mtype INTERN "grey message"
tpwrite -mtype DEBUG "black message"
tpwrite -mtype NOTE "blue message"
tpwrite -mtype OK "green message"
tpwrite -mtype WARNING "yellow message"
tpwrite -mtype ERROR "red message"
```

> Caution:
>
> Grey log lines (created using the switch `-mtype INTERN`) are not used with the internal log index and therefore they are not found with the log search functionality in the T4x.
>
> We do not recommend to write into the default log file. In order to write to another log file than the tpapps log file, add the parameter `-logchannel` to the command `tpwrite` with the info where to write to.
>
> There are more values for the parameter `-mtype` but these are used internally only, please do not try that.

The parameter **-stacklevel** will indent the message text in the log file. For example the call `tpwrite -stacklevel 3 "test"` in the mapping will create six blanks (double the number given as stacklevel) followed by the actual message.

Session Log

The T4x Session Log is one log file for every Teamcenter session. So whenever Teamcenter is restarted, a new T4x session log file is created. Use `-logchannel [::T4X::CORE::getSessionLogChannel]` to write into the Session Log:

```
tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] "+++ test"
```

User Log

You can define a custom log channel and write your log information to it:

```
tpwrite -logchannel "tmp/Custom.log" "+++++ test +++++"
```

> Caution:
>
> When writing to a custom logfile, do not try any file outside the *tmp* log folder because you may not find it.

Transaction Log

This is a special log file that is created for each T4x Transaction. It contains an overview of the transaction with the result of the different steps of a standard transaction:

- Reading data from Teamcenter.
- Processing the data in the T4x mapping and send the data to the ERP system.
- Processing the ERP system answer on this data package.
- Processing the data in the T4x reverse mapping and store the data appropriately into Teamcenter.

In order to write your own messages into a Transaction Log file, use the following commands in the functions for the mapping to the EA system. For better visibility, it allows creating the info in red, yellow blue and green:

```
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId ERROR "T4x test
red"
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId WARNING "T4x test
yellow"
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId INFO "T4x test
blue"
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId NOTE "T4x test
green"
```

For the reverse mapping, replace `writeCustomMappingLog` with `writeCustomReverseMappingLog`.

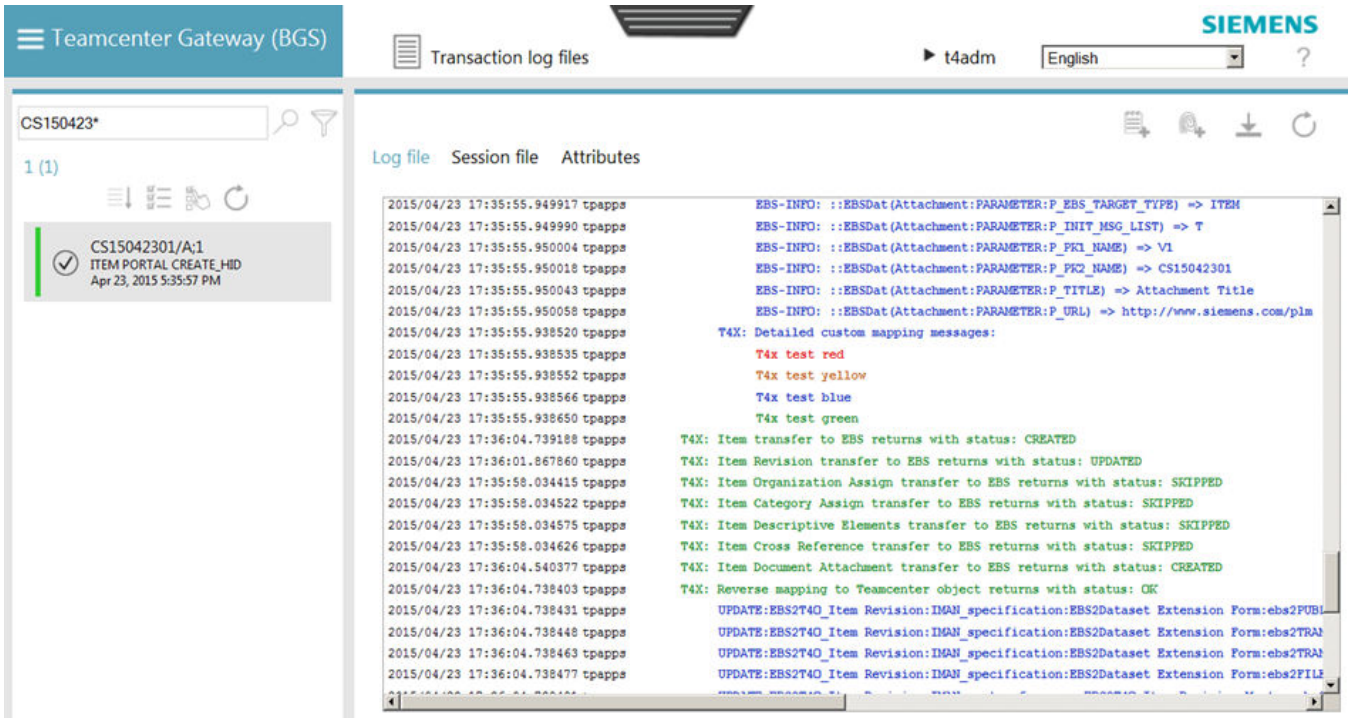
These calls need to be different in order to put them into the correct section in the log files. So if the call `writeCustomReverseMappingLog` is put in the standard mapping function `TC_Object2SAP_*`, the output is created still in the reverse mapping function.

Caution:

In a productive environment, we recommend putting only minimum log information into it. Only important Information should be stored like "Please fill the mandatory field...". For further information prefer the output with `tpwrite` to a session or user log. Or if needed, add a **Log Attachment**.

The parameter values for the color are not all the same like for `tpwrite`.

Transaction Log files can be accessed via the Gateway Menu (Portal transactions only) or in the Basic Gateway Service Admin UI:



6.7 Write to the T4x Workflow logfile Dataset

T4x offers the rule handler `T4X-attach-LogfileDataset` for creating an additional Workflow log file that will be attached to the Workflow as an attachment. In case of an error, the error message defined with `storeMessage2` will be shown there. For writing into this file no matter if an error occurred, use the function `storeMessage2DatasetLogfile` instead:

```
::T4X::CORE::storeMessage2DatasetLogfile "MM Mapping" "Test message"
```

- The parameters for the function `storeMessage2DatasetLogfile` are the same as for `storeMessage2`.

- The main difference between these two functions is that `storeMessage2DatasetLogfile` writes to the Workflow logfile Dataset.
- In any case, the message created with `storeMessage2DatasetLogfile` is written to the T4x session log file as well.
- If the parameter `Type` is set to `ERROR`, the message is:
  - in the Teamcenter Gateway session log file marked with `T4X-ERROR` (shown in red),
  - and it is written into the *apps_error.log* file additionally (shown in red, too).

7. Technical Configuration

7.1 TCL Basics

T4x provides a simple programming environment based on TCL for the mapping of data between the systems (Teamcenter and system you have to connect with). TCL is a widely used scripting language; a TCL interpreter is integrated into the T4x infrastructure. A very useful help on TCL can be found on the following Internet page:

Internet page **http://tcl.tk/man**

(quick command reference **http://www.tcl.tk/man/tcl8.5/TclCmd/contents.htm**).

Please see the following chapters for more details:

- **TCL Procedures**
- **Output Functions**
- **Namespaces**
- **Variables**
- **List Variables**
- **Data Arrays**
- **Some Important TCL Functions**
- **Traps to avoid in TCL**

7.1.1 TCL Procedures

The TCL interpreter comes with many built-in procedures that can be used inside a TCL program. Those procedures include:

- string functions,
- arithmetic functions,
- basic input and output etc.

Also a number of T4x functions (e.g. log output, configuration file access, connection, ITK handling, etc) are available as TCL procedures. More functions can be defined as TCL procedures on demand.

Any TCL procedure has to be defined with a unique name and a list of arguments:

```
proc Person_Data {Name Street City} {
```

Then the argument variables Name, Street and City have to be passed by the calling routine and can be accessed inside this procedure. It is possible to define procedures with the same name but a different number of arguments, but there is no "function overload" as known from other programming languages such as "C". TCL will always use the last definition it read. So defining the same function name twice in the same namespace is not recommended.

A function does not need a `return` but it may contain several `return` entries wherever necessary in the code. A `return` causes the function exiting at this point and returning the specified value to the calling routine. A return value is recommended but not needed. So the last command in every function should be for example `return 0`.

To call this function, type for instance:

```
Person_Data "Name1" "Stree1" "City1" #or even without quotation marks:

Person_Data Name1 Street1 City1
```

In both cases, those three strings (they are not variables) are passed to the function `Person_Data`. Calling a function with a wrong number of parameters will result in an error message `wrong # args`. If you are not sure how many arguments you want to pass to this function, you can add the key word `args` to the argument list in the function definition. If you define the function as

```
proc Person_Data {Name Street City args} {
```

you may call it with more parameters without getting an error. Using less than three parameters in the function call will result in the `wrong # args` error again. If you provide more than three, than they will be put into `args` which is a "list variable", see below.

Another way of calling a function is to put its name in square brackets. Then its return value (maybe an empty string if the function has no return value) is inserted at the point in the code where it was called. The square brackets include the function name and the parameter list. For example, if the above mentioned function has a return value `OK`, the call

```
tpwrite "call to Person_Data: [Person_Data Name1 Street1 City1]"
```

will result in the output `call to Person_Data: OK`. Of course the function will be executed in the same way as above and if there are output commands those will also be returned.

7.1.2 Output Functions

The standard TCL command for an output is `puts`. Puts is used for output to files and to console channels as well. For output in T4x, only `tpwrite` is used which is an adaptation of the TCL standard `puts` modified specifically to write into the T4x log files.

In most cases, you should not use the standard TCL command `puts` in any of the T4x mapping files because you will not find the output in the logs. "puts" will only create an output as expected when not writing to a file, e.g. in a T4x test script, where the output is sent to the created web page, but not stored in any of the logs. So in most cases, you should prefer `writeCustomMappingLog` (or `writeCustomReverseMappingLog`, respectively), that sends its output to the web page and to the transaction log file, see below.

T4x will create each log file (always within the log path) without warning if it does not yet exist (and also create the directories that do not yet exist).

In order to store additional debug output in a log, you may just use the following line in any of the mapping files: `tpwrite "++++++++++ output text"`

When stated without options, this writes `"++++++++++ output text"` to *tpapps.log* (depending on the T4x version the log file name is *tpapps32.log* or *tpapps64.log*). As the log files maybe very big, we recommend adding a string that is found easily, in the above example the plus signs.

If you want to write anything into a specific file, add the option `-logchannel`:

Example to write a line to the file *tmp/test.log*:

```
tpwrite -logchannel "tmp/test.log" "+++ test +++"
```

Example to write a line to the session log file:

```
tpwrite -logchannel [::T4X::CORE::getSessionLogChannel] "+++ test +++"
```

In addition to `tpwrite` there are two more functions to write to the transaction log file:

- `::T4X::TRANSLOG::writeCustomMappingLog` (writes a line to the mapping section of the transaction log file)
- `::T4X::TRANSLOG::writeCustomReverseMappingLog` (writes a line to the reverse mapping section of the transaction log file)

Depending on the key word `ERROR`, `WARNING` or `INFO`, the output will be shown in red, orange, or blue.

Examples to write to the mapping section of the transaction:

```
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId ERROR
```

"++++++++++ Test Error Text"

```
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId WARNING
```

"++++++++++ Test Warning Text"

```
::T4X::TRANSLOG::writeCustomMappingLog $TransactionId INFO
```

"++++++++++ Test Info Text"

Examples to write to the reverse mapping section of the transaction:

```
::T4X::TRANSLOG::writeCustomReverseMappingLog $TransactionId ERROR "Error
Text"
```

```
::T4X::TRANSLOG::writeCustomReverseMappingLog $TransactionId WARNING
"Warning Text"
```

```
::T4X::TRANSLOG::writeCustomReverseMappingLog $TransactionId INFO "Info
Text"
```

7.1.3 Namespaces

Namespaces are a kind of grouping mechanism for functions and variables. As long as you are in the same namespace you can access every function and variable by its name alone. When trying to use it from inside another namespace, you have to call it with the whole namespace identifier. In the mapping files, there is always a section starting with the word namespace.

This is at first the name of the current namespace (e.g. for material mapping:

`namespace eval ::T4x::CUSTOM::MAPPING`), then the names of the functions that are allowed to be accessed from outside this current namespace (namespace export...). In fact, a function may be called from within another namespace (in the T4x context this is mostly from within another mapping file) even without the namespace export definition when the call contains the complete namespace, e.g. the call to the object (Product or Material etc.) mapping function would be

```
::T4x::CUSTOM::MAPPING::TC_Object2YourSystem_Object.
```

The closing bracket of the `namespace eval` is at the very end of a mapping file, i.e. the namespace definition contains all the functions defined in that file. If a function inside a namespace is not stated with `namespace export`, it is not possible to call it from outside this namespace.

This entire name is always stated in the function header as "Method".

Please note that:

- Any variable defined outside a namespace and outside a function is accessible everywhere without a namespace name.

- The key word "variable" within a `namespace eval` command creates and initializes a namespace variable, e.g. `variable test "abc"`. If this command (but without the initializing "abc") is called from within a procedure inside this namespace, it will refer to the same variable `test`. For instance, in the BOM mapping file, the command `variable BomPositionInfo` does not create a new variable, but it allows the access to the variable `BomPositionInfo` that is created in the same namespace but outside the function definition. Without the `variable BomPositionInfo`, you would have to write it with the whole namespace name to access it.

7.1.4 Variables

Variables have to exist when they are called, else an error `can't read <name>: no such variable` is given. You can introduce a variable at any point of a mapping file by simply defining it, e.g. `set Var1 "value"`. Then you can use this new variable, e.g. `tpwrite $Var1`.

The counterpart of `set` is `unset` (clear memory). This may be stated at any point in a program, but it is never necessary from the logic point of view. It might be useful for memory size reasons.

It is not possible to concatenate strings with the plus ("+") symbol. Strings can be built up by just putting its parts into quotation marks, no matter if they are variables or constant strings (but constant strings always have to be put in quotation marks):

```
set var1 "$var1 and $var2"
```

If the variable var1 is rather long, then it is more efficient to use the append command:

```
append var1 " and " $var2
```

This allows any number of arguments. All of them (besides "var1" itself) are put together into the variable "var1". It does not matter if this already exists; if not it is created.

Variables are either "global" (if defined outside any procedure) or "local" (if defined inside a procedure, then they are normally only accessible inside this procedure). There are two possibilities of accessing a global variable within a procedure without defining a new (local) one:

- tell the TCL interpreter that this name in this procedure always refers to a global variable: `global paul`

- tell the TCL interpreter at the call to the variable that it is a global variable: `$::paul`

Then inside the procedure `jim`, both calls to the variables refer to a global variable:

```
set john "empty" proc jim{args} { global paul set var3 $paul set var7
$::john }
```

We recommend only using the scope operator ("::"), not the keyword global. So you always can see if a global variable is used. This also allows using a variable in more than one function.

You can put that variable in the header of the file (maybe inside or outside a "namespace" definition):

Definition:	`set ::Test_Var1 "content of Test_Var1"`
Call:	`tpwrite $::Test_Var1`
	`tpwrite "Variable: $::Test_Var1"`

That tells the TCL interpreter that `Test_Var1` is a global variable (so it should not look for that in the local variable memory space). It also works in different files; see Direct Mapping of Constants. The file names do not matter (see **Manage a development environment**) but the names of the namespaces, functions etc must not be modified.

Please note that:

- If a word in an output starts with a dollar sign `$` it is a variable, else it is a constant.
- Be careful with special characters in the source code:

  Blanks divide different words (e.g. variable names or TCL key words).

  An open square bracket `[` will expect a close square bracket and a function name (e.g. `string trim`) inside it to execute.

  An open curly brace `{` will expect a close one, too. It is used to group expressions where logically only one is allowed, e.g. in an "if" structure.

- In a string, curly braces divide a variable name from a text, so you can combine the value of a variable with other text without blanks, e.g. `(if Var1 is "123") set test_var "start{$Var1}end"` will set the variable `test_var` to "start123end". If you leave the curly braces, it will cause a TCL error if the variable `$Var1end` is not defined.
  As a rule, TCL treats any variable as a string, independent of its content. To use its numeric content, use the `expr` command, e.g. you can count something with `set Index_new [expr $Index+1]` as that sets the variable `Index_new` to the value that is one more than `$Index`.

  > Caution:
  >
  > If `$Index` is empty or contains other characters than digits, this causes an error.

- Variables can be evaluated as Boolean, too: `if {$error_occurred} { ... A value "0" (zero)` is interpreted as `false`, any other numeric value as `true`. If the string contains other characters than digits, this causes an error.

> Caution:
>
> In such a string, "000" is evaluated as zero and "001" as one.

- If the position in a variable (string, field, list...) is needed, note that the first element is always referenced as zero.

- It is possible (but unwise!) to use the same name for a global and a local variable. Although it is possible to distinguish them exactly with the scope operator `::` there surely will be a confusion at any time.

7.1.5 List Variables

A variable contains mostly one string, but there is a way to have several strings in one variable: create a list variable (this is done in TCL with the key word list). The most important commands for lists are:

- `concat arg1 arg2...`: treats each argument (any number of arguments allowed) as a list and concatenates them into a single list. It also eliminates leading and trailing spaces in the arguments and adds a single separator space between them.

- `lappend`: append an element to a list variable.

- `llength`: count the number of elements in a list.

- `lindex`: retrieve an element from a list.

7.1.6 Data Arrays

Arrays are a construction where several variables can be accessed with a common variable name and one or more indices. The indices are not integers (as known from other programming languages) but strings. This means there is no unique order of the indices and not all possible variables with the defined indices have to exist at any time. Maybe you have an array with the known indices "b" and "c" at one point of a TCL program. If you later find the same variable name with the index "c", it does not mean necessarily that the same variable name with the index "b" exists. So if you find the error message`can't read ::TcData(ItemInfo:ObjectClassName)no such variable` in the test mapping (in the Admin UI) it is in most cases not a programming error, but this variable was not created because the test mapping tried to get data from an object that does not exist.

For further information on array handling in TCL, see the TCL tutorial:

http://www.tcl.tk/man/tcl8.5/tutorial/Tcl22.html

7.1.7 Some Important TCL Functions

- `[binary format {H*} $hexVar]`

Gives the clear text of the binary coded variable `hexVar`. Although in TCL, "binary format" is mainly used for creating a binary string from normal data, binary format `{H*}` will do the opposite and extract the readable text data from the binary format.
T4x additionally offers an own function called `tpco_formatHEX16` which does the same in principle (de-hex a text in hex format) but it has the advantage that it does not end with a runtime error if the input string is not a pure hex string, but it just returns an empty string. The original TCL command will end with an error message like:
expected hexadecimal string but got "48656c6c6fxyz" instead
`tpco_formatHEX16` can also correctly handle hexed multi-byte characters.

- `[binary scan $var {H*} hexVar]`
  Performs a hex-encoding of the value of variable `var` and assigns it to variable `hexVar`. T4x additionally offers `tpco_scanHEX16` with a similar functionality (convert plain text to HEX16) but with the advantage that it can encode multi-byte characters correctly.

- `[clock format [clock seconds] -format "%Y-%m-%d %R"]`
  Gives a formatted time string, here for example:
  "2015-03-01 14:32" "[clock seconds]"
  is the number of seconds elapsed from a virtual start time
  Note that according to ISO 8601 specifications, the TCL clock functions are defined for the time range from year 1970 to year 2037 only.

- `[format "%s.%s" $Num1 $Num2]`
  Generates a formatted string in nearly the same way as the ANSI C `sprintf` function (see the above mentioned TCL help for details).

- `[info …]`
  Return information about the state of the TCL interpreter, e.g.
  `[info exists varName]`
  returns "1" if the variable named `varName` exists in the current context (either as a global or local variable) and has been defined by being given a value, returns 0 otherwise. Useful to test if a variable exists.

- `regsub -all {\.} $Quantity {,} Quantity_new`
  Copies the string `Quantity` to the string `Quantity_new` with all the dots replaced by a comma. The variable names may be the same (then it is overwritten).
  Please see the point regular expressions in the next chapter.

- `[scan $string1 $format result_strings]`
  Parse string using conversion specifiers in the style of `sscanf` from C++, e.g.
  `set parts [scan $inputstring "%\[^_\]_%s" v1 v2]`
  Searches for an underscore ("_") in the variable `inputstring` and saves the two parts (one before and one after the underscore) into the two variables `v1` and `v2`. If there is no underscore, `v2` will be empty and `v1` the same as `inputstring`. As this is a very powerful function you should read the manual when using it for special needs.

- `[string length $Var1]`

Gives the number of characters in `Var1`.

- `[string range $Var1 0 9]`
  Gives the first 10 characters of `Var1`.

- `[string repeat " " 5]`
  Gives a string that consists of five blanks.

- `[string toupper $Var1]`
  Converts `Var1` to upper cases.

- `[string trim $Var1]`
  Cuts "white space" (spaces, tabs, newlines, and carriage returns) at beginning and end of `Var1`.

- `[string trimleft $Var1]`
  Cuts "white space" at beginning of `Var1` (analogue: `trimright`).

- `[string trim $Var1 "x"]`
  Cuts every "x" at beginning and end of `Var1`.

  > Caution:
  >
  > If you have a string like "abcddde" and perform `[string trim $Var1 "de"]` on that, you will not get "abcdd", but "abc".

- The most important TCL operators for numbers are == (equal), != (not equal), < (smaller), > (bigger); for strings: `eq` for equal, `ne` for not equal.

- In TCL, the standard `if` command is like the following example:

```
if { $var1 eq "abc"} {
puts "OK"
} else {
puts "nio"
}
```

- In TCL, the standard `for` loop is (example for numbers 0 to 9):

```
for {set x 0} {$x<10} {incr x} {
puts "x is $x" }
}
```

7.1.8 Traps to avoid in TCL

- Every open bracket needs a close bracket of the same type (parenthesis "( )", curly brace "{ }" or squared bracket "[ ]"). Many text editors allow searching for a corresponding bracket.

- The open curly brace has to be in the same line as the command that opens it, and it needs a blank before it. A line that only contains an open curly brace (as often used in C programs) is an error in TCL. A correct line in TCL is:
  ```
  if {$Status == "ERROR"} {
  ```

- When commenting out an open curly brace, there has to be a corresponding close curly brace that is commented out, too. This example would cause an error (even if there is no corresponding close curly brace in the source code):
  ```
  # if {$Status == "ERROR"} {
  ```
  This is correct (no matter if there is code – commented or not – between the two lines):
  ```
  # if {$Status == "ERROR"} {
  # }
  ```

- Do not add any comments in a switch structure. Even inside a comment, the logical structure is checked and even the code may be used as if it was not commented:

  This example does not create a runtime error, but nevertheless the result will be "2" if the checked variable has the value "EBS OK":
  ```
  switch -exact -- $Value {
  # Test: {EBS OK} {set xy "2"}
  {EBS OK} {set xy "1"}
  {EBS NOT OK} {set xy "0"}
  default {set xy "error"}
  }
  ```

  Although it looks more logically correct, a line like that in the "switch" will cause an error:
  ```
  # {EBS OK} {set xy "2"}
  ```

  So please move any comment outside the "switch" loop.

- The backslash indicates that the following control character is not to be used as such. So if a source code line ends with "\", the following line is treated as if it was in the same line because the "new line" character is ignored. If there is a blank following the backslash it will not work (as the blank is ignored which does not mean anything).

- Some characters are evaluated as "regular expressions", i.e they have a special meaning. That's why the backslash in the first curly braces of the following command is necessary (copy the string `Quantity` with all dots replaced by commas to the new string `Quantity_new`):
  ```
  regsub -all {\.} $Quantity {,} Quantity_new
  ```
  For details, see TCL online help: **http://www.tcl.tk/man/tcl8.5/TclCmd/re_syntax.htm**

- Comments after a statement are not allowed. The only way is to tell the TCL interpreter to handle the following as a new line. This is done with a semicolon (";"). The following example is a correct line with a statement and a comment in only one source code line (but internally treated as two lines):
  ```
  puts "Hello world" ;# This is a comment after the command
  ```

- TCL does not handle variables with a defined type as known in other programming languages. In principle, every variable is a string but TCL will try and handle it as the type it thinks it is, e.g. an octal number:

  If a string contains a number, TCL will try and evaluate it as number. Although it might seem strange, the following conditions are always true:
  ```
  if {"0" == "000"}
  if {"+00001.2" == "1.200"}
  ```

  For the same reason, the equation with two strings containing an "e" may fail as TCL will treat it as a number in exponential notation. In contrast to integer, the numeric type "float" (floating point number) does not have a defined overflow, so a number with a too high exponent will not be treated as another smaller number, but as value "infinite". So in the following example, TCL will compare "infinite" with "infinite" which means they are equal:
  ```
  if {"5e321" == "6e321"}
  ```

  To avoid this equation problem, be sure to tell TCL you want to compare strings instead of numbers (`eq` for "equal, `ne` for "not equal"):
  `if {"5e321" == "6e321"}` is true
  `if {"5e321" eq "6e321"}` is false.

  Any string that may be a number and starts with zero is treated as an octal number, so a string "08" or "09" leads to a TCL error when it is used in an equation. If you read strings that should be evaluated as numbers, be sure that there are no leading zeros.

- TCL allows error suppression with the command `catch`. Then the program continues after the line that caused the error and no error message is stated. As this may prevent error tracking, you should only use it if you cannot find another way. Of course it is better to make a program fail-safe by detecting and handling errors systematically instead of only preventing error messages.

- If a function with the same name is defined more than once, TCL uses the last one that it reads. This may happen if you have backups of your mapping files in the directory **<T4O_ROOT>\var\mmap\<t4x>_mapping_config** (or **<T4O_ROOT>\var\mmap\t4x_mapping_config** respectively), so avoid this even if the file names are different.

7.2 Mapping basics

Direct mapping of constants

This functionality is provided to allow the usage of the same variable value in more than one mapping file.

A useful example is the SAP plant ID which may be used for the MaterialMaster and for the BOM as well. Store it as a variable in the settings file **<T4x_ROOT>\var\mmap\t4s_mapping_config\t4s_mapping_config.sd** and use it in both of the mapping files.

Example:

```
set ::T4S_Defaults(PlantId) 1000
```

Usage for MM:

```
set ::SAPDat(Material:MarcData:PLANT) $::T4S_Defaults(PlantId)
```

Usage for BOM:

```
set ::SAPDat(Bom:BomHeader:Plant) $::T4S_Defaults(PlantId)
```

Both of those calls will get the same value for the `PlantId`. So that value has to be modified at that one point of the mapping only, no matter in how many mapping files it is used.

You may introduce any variable you like in this style.

Caution:

- Be sure not to set there any variables that T4x will modify at runtime automatically, e.g. the whole variable array `::StatusInfo`; it is altered in each transaction by T4x itself.
- If you define an own global variable (i.e. its name starts with the double colon, e.g. `::TestValue`) be sure to delete it between two runs because it may keep its previous value. Depending on the way it is used this cleaning should be done in the beginning or the end of the transaction: `unset ::TestValue`.

Obtaining special Teamcenter and T4x information

There is some Teamcenter information that is available in the mapping independently from the data type. It is stored in the ::TcData array and the value is encoded in hexadecimal. Here are some examples:

- `::TcData(GatewayMode)`
- `::TcData(ItemInfo:ObjectClassName)`
- `::TcData(SiteIdName)`
- `::TcData(UserId)`
- `::TcData(UserName)`
- `::TcData(WorkflowJob:owning_user)`

Example for getting the current Teamcenter database name:

```
if {[info exists ::TcData(SiteIdName)]} {
  set SiteName [string trim \
  [::T4X::TC::MAPPING::decodeTcDataString $::TcData(SiteIdName)]]
} else {
  set SiteName "SiteName_not_found"
}
```

Example for getting the current Teamcenter user name and user group name:

```
set current_TC_user  [::T4X::TC::MAPPING::decodeTcDataString \
                     $::TcData(UserName)]
set current_TC_group [::T4X::TC::MAPPING::decodeTcDataString \
                      $::TcData(GroupName)]
set current_TC_role  [::T4X::TC::MAPPING::decodeTcDataString \
                      $::TcData(RoleName)]
```

You can find a more complete list of this kind when executing the test mapping script in the Admin UI of the Gateway Service and checking the data output.

Please note that:

Caution:

You can´t always use all variables in the mapping exactly the same as shown in a test mapping, but some of those data. In order to use something you find in the test mapping output, try as follows:

- Getting the hexadecimal value:

```
set ObjectTypeName $::TcData(ItemInfo:TypeName)
```

- Convert the hexadecimal value to correct UTF-8 character encoding:

```
 set TC_user [::T4X::TC::MAPPING::decodeTcDataString
$::TcData(UserName)]
```

All the data that are directly related to the queried Teamcenter object can only be accessed by the `FieldMapping` function, or in BOM mapping with the `IndexedFieldMapping` function, respectively.

Read Teamcenter data in hexadecimal format or without trimming blanks

By default, the functions `FieldMapping` (for reading a single object from Teamcenter), `IndexedFieldMapping` (for reading object arrays from Teamcenter, e.g. BOM data) trim away leading and trailing blanks from the strings they read from Teamcenter. Example: There is a value " A" stored in Teamcenter, but T4x will cut the blank away and use only the "A" instead. Furthermore, functions

`FieldMapping` and `IndexedFieldMapping` have a switch for defining if the value is returned in hexadecimal format or not. By default, this is set to `false`. If you need to change this default behavior to return the read values in a hexadecimal format or without trimming leading and trailing blanks, add the following optional parameters to the call. The complete function signatures are as follows:

```
proc FieldMapping {FormName FieldName {UseBinaryFormat FALSE}\
  {TrimValue TRUE} args}
proc IndexedFieldMapping {Index FormName FieldName {UseBinaryFormat
FALSE}\
  {TrimValue TRUE} args}
```

So if you want to keep the blanks from the Teamcenter data, modify the call, e.g.

from `set xy [::T4X::TC::MAPPING::FieldMapping $Form MyValue]`

to `set xy [::T4X::TC::MAPPING::FieldMapping $Form MyValue FALSE FALSE]`

Set the parameter `UseBinaryFormat` only in very special cases to `true`. Please note that it is impossible to set the parameter `TrimValue` without setting `UseBinaryFormat`.

Read up to 10 decimal places from Teamcenter double precision properties

By default T4x only reads up to 6 decimal places (e.g. "0.123456"). This number can be configured from 5 up to 10 digits after the decimal sign.

Define the site preference `T4X_NoOfDecimalDigits4DoubleExtration`of type `Integer` and `Single` value. Enter a value greater than 4 and lower than 11.

Evaluate a Boolean attribute in a Teamcenter form

There are three possibilities when you query a Boolean attribute in a Teamcenter form, but in most cases there is no need to distinguish between all of them explicitly:

- Attribute does not exist or is empty
- False
- True

The first approach:

```
set flag_1  [::T4X::TC::MAPPING::FieldMapping … test_att]
if {$flag_1 == "1"} {
  tpwrite " yes"
} elseif {$flag_1 == "0"} {
```

```
  tpwrite " no"
} else {
  tpwrite "Could not evaluate test_att"
}
```

If you just state `if {$flag_1}` in the code, you will get a TCL error if the attribute does not exist, because this variable will be empty. The Boolean evaluation in TCL works with numbers, where a value "0" (zero) is interpreted as `false` and any other numeric value as `true`. If the string is empty or contains other characters than digits or internal boolean codes (`false` or `true`), that would cause an error.

> Caution:
>
> in such a string, "000" is evaluated as zero and "001" as one.

The correct way is:

```
if {[::T4X::TC::MAPPING::testFieldExists \
"$ItemRevisionType:IMAN_master_form_rev:$ItemRevisionType Master"
test_att]\
 eq "OK"} {
  tpwrite "+++++ test_att exists"
  set flag_1  [::T4X::TC::MAPPING::FieldMapping …  test_att]
  if {$flag_1} {
    ...
  }
} else {
  tpwrite "+++++ test_att does not exist"
}
```

Read Teamcenter form attributes of type "string array"

```
set StingArrayContent [::T4X::TC::MAPPING::FieldMapping \
  "$ItemRevisionType:IMAN_master_form_rev:${ItemRevisionType} \
   Master" "Keywords"]
set StringArrayList [split $StingArrayContent "\n"]
set Value1 [lindex $StringArrayList 0]
set Value2 [lindex $StringArrayList 1]
```

The following example shows how to read the values stored in the IRM for the attribute `Keyword` with type `string array` (also known as "multi-valued attribute"):

Modify the T4x encoding

By default the T4x tries to identify the correct character encoding that is used by Teamcenter for the data access automatically, but If you should encounter problems that T4x does not seem to process special characters correctly, set the following key additionally in *<t4x>_mapping_config.sd*:

Example:

```
set ::T4X::TC::MAPPING::CtrlInfo(useTcCodePage)  cp1252
```

In a very special case maybe the value "utf-8" is better than the here shown "cp1525".

In fact, the problem is not how the T4x talks with the EA (you have to make sure you transfer data with the correct encoding in your custom code) but how it gets the data from Teamcenter, so you have to ask Teamcenter for the correct setting. In order to find correct value for this installation, do the following:

- Find the encoding set currently used by Teamcenter from the TC-Server syslog file look there for the following line: `Session is running in codeset 1252 17` or check the value of the `::TcData(SystemEncodingAliasList` variable within the mapping or during a mapping test script run. The output will look similar like in this example:
  `TcData(SystemEncodingAliasList)=ibm-5348_P100-1997 ibm-5348 windows-1252 cp1252`

- Find the same T4x encoding by checking from the T4x command shell:
  - type `tps` <Enter>,
  - the prompt will change to a percent sign,
  - type `encoding names` <Enter>,
  - a list of all supported encoding types is shown,
  - find the corresponding name from the list (in this example, it is `cp1252`),
  - type `exit` <Enter> to change back to the standard prompt.

- Set the following line (the value is what you found in that list; needs to be written exactly as shown in the list, note the lower case letter c) in the file **<T4x_Root>/var/mmap/t4x_mapping_config/t4x_mapping_config.sd**, e.g.:
  `set ::T4X::TC::MAPPING::CtrlInfo(useTcCodePage) cp1252`

> Caution:
>
> If you think Teamcenter Gateway does not handle some special characters correctly that are written explicitly in the mapping files (may occur especially with Chinese characters), be sure to not include the actual character but the so called "slash-u code" instead because many *text editors* do not handle those characters correctly internally although showing them correctly.

Example: You want to hard code the string "中文" in one of the T4x mapping files.

Instead of putting it there as such, use its slash-u code (\u4E2D\u6587 in example bellow):

`set Description "$Desc 中文"` avoid that `set Description "$Desc \u4E2D\u6587"`

To find the correct slash-u code, do the following in the T4x command shell:

- start tps (see above: type `tps` <Enter>),
- define the procedure `getUnicode` (the blanks do not matter, it is just for the optical logic in this description here),
- call this new procedure with your Unicode string and find the slash-u codes:
  `C:\Temp> tps`

```
% proc getUnicode {string} {
 set rv {}
 foreach c [split $string {}] {
 scan $c %c c
 append rv {\u}
 append rv [format %.4X $c]
 }
 return $rv
}
```

```
% puts "check unicode [getUnicode 中文]"
```

Output: `\u4E2D\u6587`

7.3 Manage a development environment

Of course testing is an important part of the implementation. It is highly recommended to have at least one separate Teamcenter installation for testing the T4x use cases – instead of using the production environment. It is best practice to have one installation per developer (development environment), one for testing and one production installation. The same concept should be used for the enterprise application, at least if T4x writes data to it. For the enterprise applications it may be sufficient to have just one combined development/test environment and the productive installation. This staging concept ensures that immature T4x mappings cannot cause any harm in the productive databases and that the development phase is not hampered by requirements of the productive processes.

During the implementation phase of the T4x mappings, you can use the mapping test scripts from the T4x script environment to review the results of your mapping without actually transferring anything. There are also scripts to start complete transfers or imports. These can also be used to do mass or batch transactions using the mappings you have configured.

Steps for modifying the T4x mapping:

1. Set Teamcenter preferences to configure the Teamcenter objects whose data T4x should read from Teamcenter or may write to, respectively.

2. Specify the mapping (define the handling of the data, e.g. which Teamcenter attribute gives data to which EA field and vice versa) in the mapping files.

3. Create the compilation of all mapping source files: **<t4x>_mapping_config.rfdt**.

4. Move this file to **<GS_ROOT>\lib**.

5. Restart T4x.

You can use the template file **<GS_ROOT>\etc\<t4x>_build_mapping.bat.template** to compile the mapping and move the *.rfdt* file to the lib directory in one step. In order to use it from the operating system directly:

- In Windows: Just remove the ".template" from the file extension.
- In UNIX/Linux, define it as an executable file (the file extension does not matter).

Or use the "mapping hot deploy", see below.

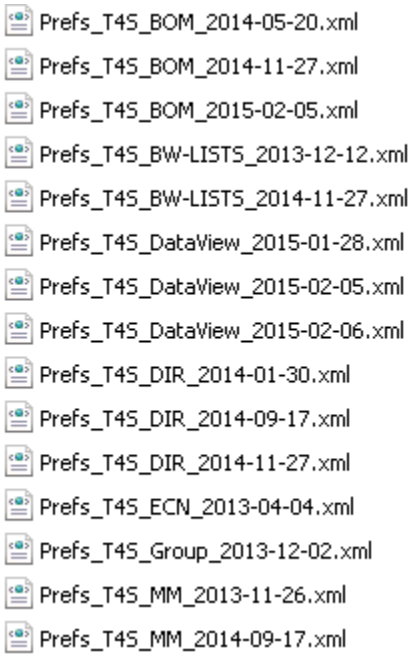
Generally, there are three kinds of data T4x can read from Teamcenter:

- Any attribute that is stored in a Teamcenter form, e.g. the standard Item Revision Master Form.
- The properties of an object, e.g. `item_id`, `item_revision_id`, `object_desc`, release date...
- Classification data as defined in the Teamcenter classification application.

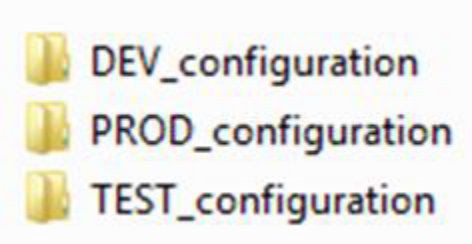
All definitions for the mapping have to be done in the T4x mapping source files. By default, every one of them contains all the definitions for one SAP object type and is named with that type, see following list. All the mapping files T4x has to use have to be located in the directory **<GS_ROOT>\var\mmap\<t4x>_mapping_config**. For more information about the content of the mapping files T4x is shipped, see the product specific Configuration Guide.

File and data management during the implementation

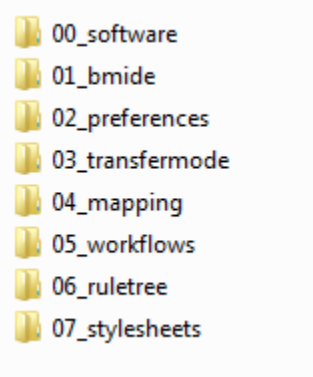
- Manage T4x preferences in separated XML files with change date.
- Do not edit preferences in TC but in the XML files and import them afterwards.



- Configuration should be saved on a network drive with regular backup.
- Configuration can be saved separately for each environment.

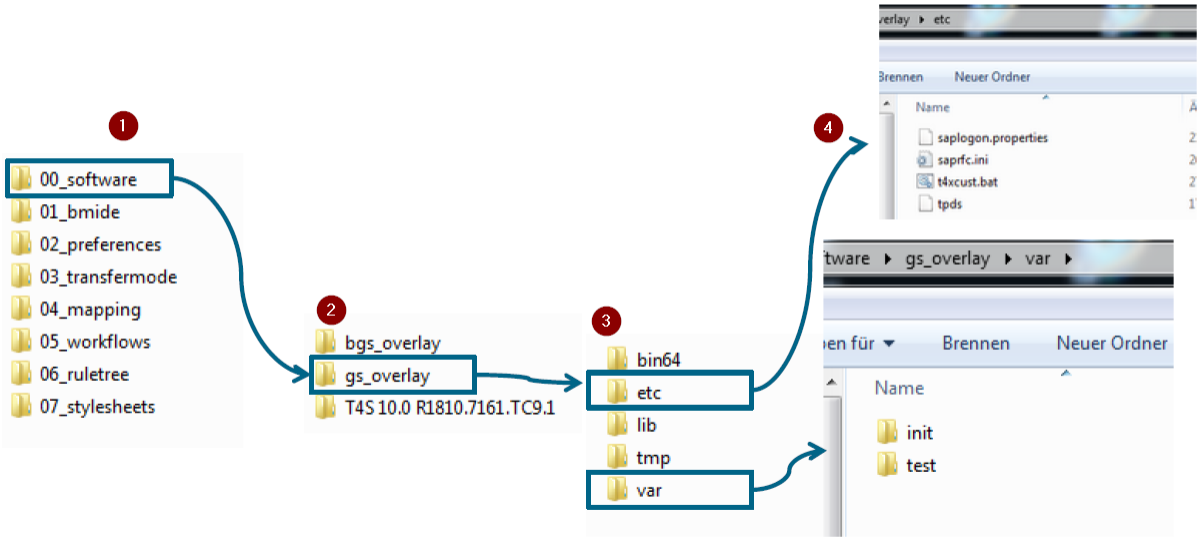


- Create a sub directory for each specific significant part of the configuration.



- Store the GS and BGS version which is used for the environment.

- Create the folder structure for the needed overlays.
- Store all the "overlays" files: for example test scripts, customer libs, patches, logon files, compiled mapping (rfdt) etc.



- SAP and Teamcenter alignment:
  - Define at the start of the project whether Teamcenter or SAP will be the leading system. In case of a data conflict, what system is „right".
  - SAP and TC classification should be aligned: Class and attribute name/id, LOV´s.
  - In a perfect world, Teamcenter and SAP have the same item ID´s.

    T4S can get SAP numbers.

    T4S can rename objects.

    SAP numbers should never start with zero! Same applies to TC ID´s.

How to check syntactical correctness of the mapping files

You may check the correct TCL semantic of the mapping files by sourcing them in a TCL interpreter. This may be very useful because syntactic errors (such as missing brackets) are very hard to track in the result. In the T4x command shell, go to a sub-directory of the *var/mmap* directory and type the following to check a file:

```
tps <t4x>_<TargetTypeObject>_mapping_template.sd
```

This should not state anything. In case of a syntactic error, it will state e.g. missing close-brace (which is obvious), but depending on the error, the result may be something strange like `can't read "TransactionId"`.

The fact that this comes during sourcing and not at runtime is a strong hint on a syntactic error, not a wrong variable definition or a similar error.

Please note that this check may state errors " invalid command name...".

This may be because functions may be defined in different namespaces which are only available at runtime, so it does not necessarily mean that there is an error in your mapping code.

How to keep T4x mapping files clean

In order to keep the mapping files easy to read it is a good idea to stick to some style rules:

- Indent your source code consistently to see loops easily. All mapping files delivered with T4x use two spaces indentation.
- Do not use tabs (by pressing the tabulator key) in a mapping file. These might be shown differently in different text editors and therefore lead to misunderstandings.
- Remove unused lines from the mapping, but comment any special adaptation.
- Place reasonable comments in order to be able to understand later what was modified (but be aware that curly braces in a comment may cause problems, so avoid that, see chapter **Traps to avoid in TCL**.
- Use significant names for your own variables. Customer specific variables should use the same prefix and same syntax, so do not call a variable `xy` but e.g. `found_new_Item`.
- Constants should be placed in the file **<t4x>_mapping_config.sd** only.
- Create outputs in order to see the results of your data processing steps; see chapter **Output Functions** for details.
- Except for the file **<t4x>_mapping_config.sd** (which is executed at T4x start-up only, not again in each transaction), do not place any code in the mapping files outside the functions,
- If you need the same program part at several places (maybe even in more than one mapping file), put it into a separate function. This function may be included in any of the mapping files. If you need several of such functions, we recommend placing them together into an additional mapping file. Remember to `source` this new file in **<t4x>_mapping_config.sd** as well.
- Create backups of the mapping files. We recommend copying the whole directory *var/mmap* and renaming it with a timestamp (e.g. mmap_YYYMMMDD). If you want to keep a mapping file backup (or any other file) in the original directory *mmap*, be sure not to keep the file extension *.sd*.

How to change specific mapping topics by several teams and use them together in the same T4x environment

The reason for that specific mapping handling is that a customer may want to allow different teams to modify their special T4x functionalities and use all the modifications together although they are not set in one *mmap* directory only. As that means that there may be several compiled mapping files *.*rfdt* working together (by default each T4x uses one file *_*mapping_config.rfdt* only), that functionality is called "cascading mapping files". So the different teams may modify their mapping stuff all together without copying the other modifications into the one mmap directory. All the T4x servers that are using the mapping (the main server as well as e.g. all T4x job agents) need to be restarted in order to use the modified mapping logic correctly.

Example 1: one team should handle the material mapping only and maybe several other teams should handle the other mapping topics, but no mapping topic should be handled by more than one such team, so that nobody may modify a mapping topic which is defined for another team.

Example 2: every team should handle all the mapping topics but for their specific ERP systems only (e.g. SAP), not for another ERP system which is used by other teams only. The reason for such a handling is not the different user names in the different ERP systems but the different field names of fields which are the same from user's point of view. So T4x has to use the different ERP field names in order to store the needed data into the seemingly same fields in different ERP systems.

This is how to handle that (example for T4S, in other T4x products the directory and file names may be different):

- Create a list of the teams and the differences of their allowed mapping stuff.
- Make sure that no user can handle the T4S directories from other teams, so the easiest way is that the different teams use another computer and nobody may use one from another team.

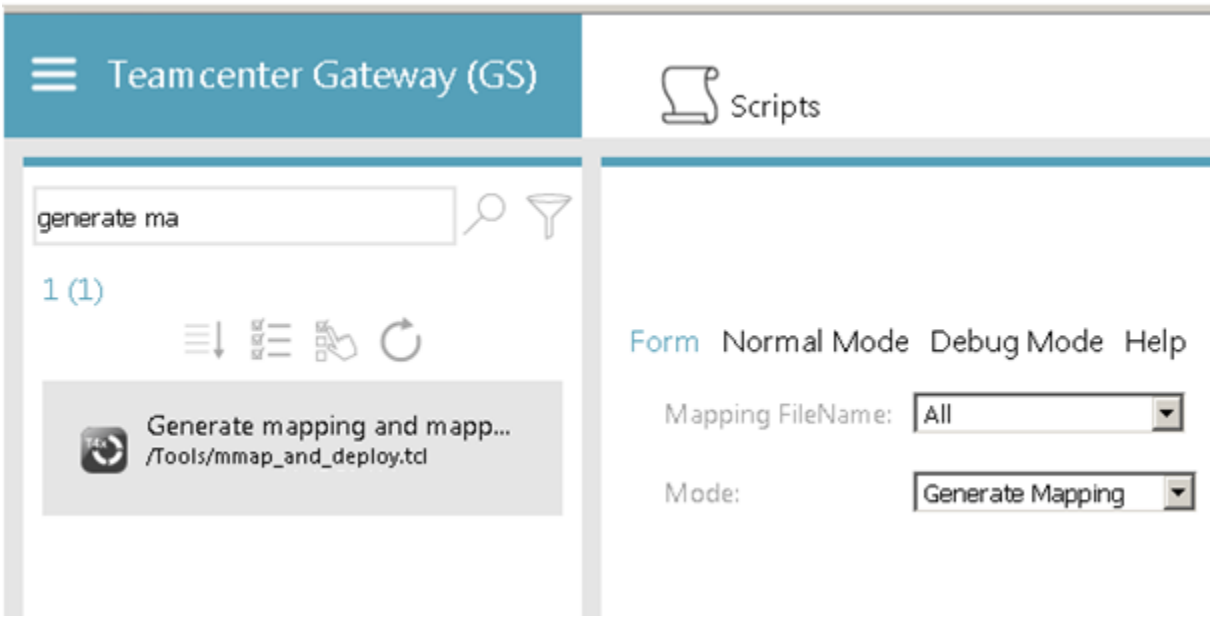
Then copy only the corresponding T4x mapping files into the directory **<GS_ROOT>/var/mmap/<t4x>_mapping_config** of the T4x installation which is used by the corresponding team.

- If there are product independent mapping files, e.g. *t4x_user_exit_template.sd*, copy them into the directory **<GS_ROOT>/var/mmap/t4x_mapping_config** instead (sub directory name *t4x_mapping_config* instead of *<t4x>_mapping_config*).
- No mapping file *.*sd* should be directly in **<GS_ROOT>/var/mmap** (without sub directory).
- In every such *var/mmap* sub directory of the different GS installations, add the file *_*mapping_config.sd* (same file name as the sub directory name) because that contains those different team settings:
  - In all those files `*_mapping_config.sd` you have to remove the lines `source -relax` for all the mapping files which do not exist anymore in that mapping directory.

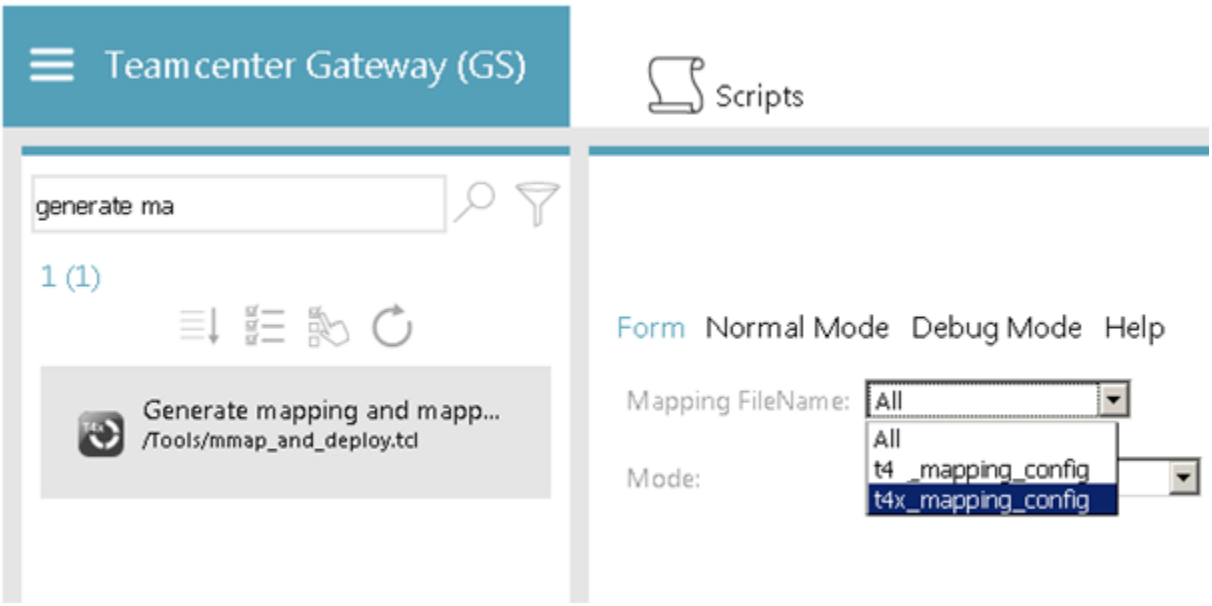
In order to load the mapping files in the current sub directory and initialize it during the T4x start-up, use the function `SYSBase::loadLibrary`. See T4CEP API Reference for details.

When a modification should be set to the used T4x product, use the script "Generate mapping and mapping deployment" in the Gateway Service Admin UI . If you follow the mentioned naming convention, both utilities will be able to generate all mapping rfdts with a single call, but it is also possible to generate a single mapping rfdt only.

All Mode by that script:



The following graphic shows the single Mode by that script, i.e. only a specific mapping sub directory is added to the actually used mapping:



And it can be used with the mmap executable, too, i.e. by running the file *<GS_ROOT>/bin64/mmap.exe* with the shown parameters in the following lines:

- All mode: just call it without additional parameters: **bin64/mmap**

All Mode with hot deployment

(i.e. the changed mapping will be used immediately without the need restarting the GS):

```
bin64/mmap -connid DEFAULT -user t4adm -passwd <yourpassword> -sdstdir lib
```

Single Mode:

```
bin64/mmap -srcdir ./var/mmap/t4s_mapping_config

-dstdir ./tmp -out t4s_mapping_config.rfdt
```

Single Mode with hot deployment:

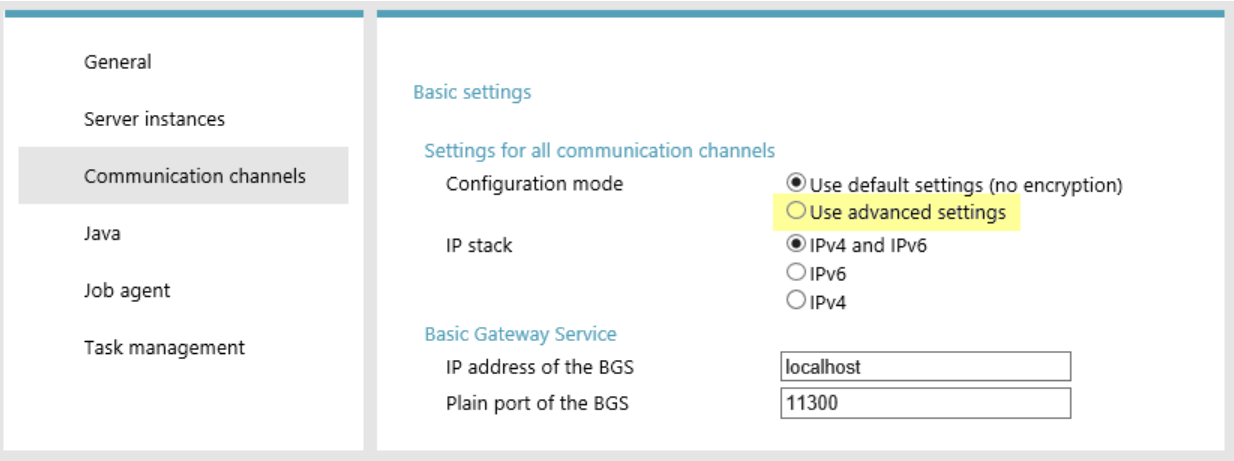
```
bin64/mmap -srcdir ./var/mmap/t4s_mapping_config

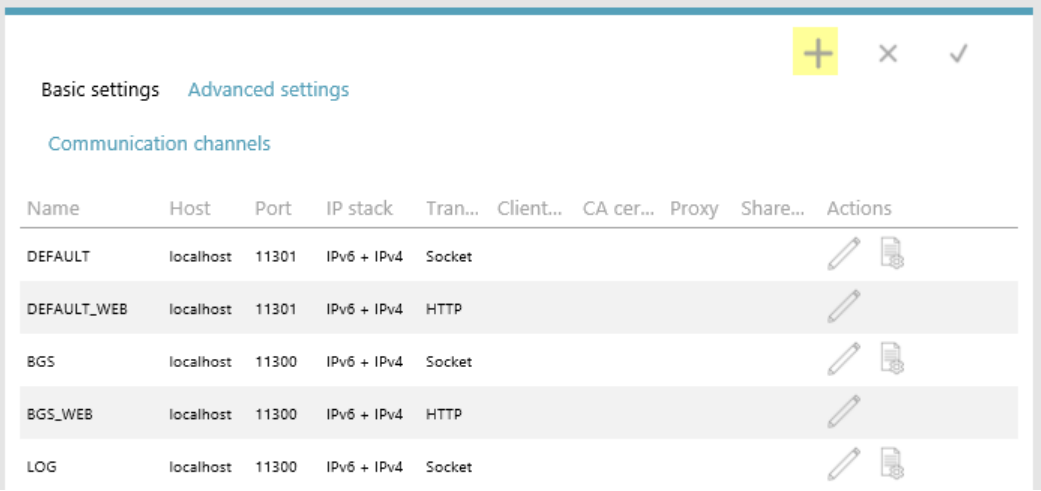
-dstdir ./tmp -out t4s_mapping_config.rfdt -connid DEFAULT

-user t4adm -passwd <yourpassword> -sdstdir lib
```

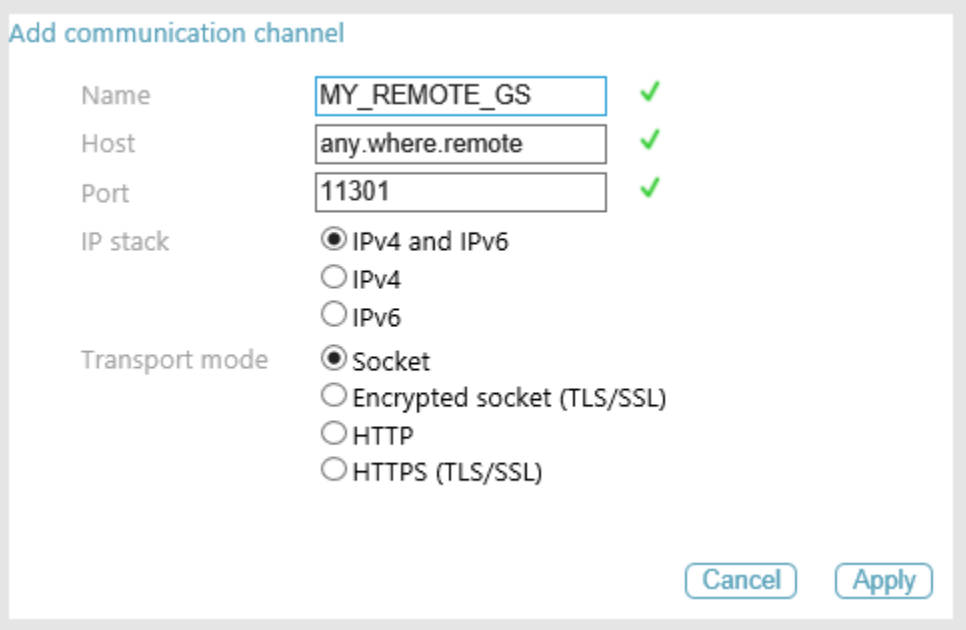
How to deploy the mapping on a remote GS using the mmap executable

During the development you can change the mapping files in a local GS installation and use this installation to generate and deploy the mapping to a remote GS. To prepare the communication with the remote GS, start your local GS, open the Admin UI and navigate to **Configuration → Communication channels**. If necessary, switch the **Configuration mode** to **Use advanced settings** to add a new channel by clicking the **Add** button in the upper right corner.





Enter a symbolic **Name** for the channel, e.g., `MY_REMOTE_GS` and provide all data necessary to establish a connection. **Apply** all the changes and restart or stop your local GS.



Now that you have created a new communication channel its symbolic name can be used in the connection ID parameter of the *mmap* executable:

```
bin64/mmap -connid MY_REMOTE_GS -user t4adm -passwd <yourpassword>
```

7.4 Troubleshooting with running Active Integration Gateway processes

If you encounter problems with any running Active Integration Gateway process, no matter if started with or without Teamcenter, check and try the following for the appropriate Active Integration Gateway server (if the process does not run at all, see previous chapter):

- Open a new command shell (without Teamcenter and Active Integration Gateway environment) and type the following:
- Copy all output to the screen:
  `set CDEB=1`
- Prevent the shell window from closing (Windows only):
  `set TP_NCONHIDE=1`
- Deactivate the "fork mode" (see above) and see the output of other libraries:
  `set TP_FORKACTIVE=1`
- Allow Windows finding the Active Integration Gateway libraries even if called without path:
  `set PATH=< T4x _ROOT>\bin64;%PATH%`
  Note that the name of the environment variable to find files without stating their path explicitly is dependent on the operating system:

- Windows: `PATH`
- Linux: `LD_LIBRARY_PATH`

- Change to the Active Integration Gateway root directory:
  `cd /d <T4x_ROOT>`
- Start Active Integration Gateway with the default port number:
  `bin64\tpapps var/init/start.apps INTERN`
- The values of the Active Integration Gateway environment variables (`TP_NCONHIDE`, `CDEB` etc.) do not matter; Active Integration Gateway just distinguishes if they exist or not.

  Caution:
  - Do not just execute bin/restart, because this does not create a visible output.
  - The name of the called executable (here: tpapps) must correspond to its start script (here: start.apps); for the BGS it is tpbgs... start.bgs, see following example.

- Try the same with BGS (change to *< T4x _BGS_ROOT>* before):
  `bin64\tpbgs var/init/start.bgs INTERN`
- Check the error message.
- If Windows shows a DLL related error message (e.g. `Entry point ... in ....dll not found`) instead of starting a Active Integration Gateway process, this may be because the stated DLL exists in one of the Windows system directories (in most cases in *Windows\system32*). If this older DLL is no longer needed, you could rename it and try again.
- UNIX/Linux environment only (maybe you need admin rights!):
  - If there is a lib related problem (something like `file ….so not found`), you can use the system tool ldd to check which libs are used by an executable. To show the used libs together with their paths, do the following in a command shell:
    - `ldd <executable>`
    - If the lib is not found in the system (what may be the reason for the problem that T4x cannot start properly), it will output something with "not found" instead of the path. So the following command will help you finding only the used libs with the word "not" in the output:
      `ldd <executable> | grep not`
      Example (replace <T4x> with your product-specific name): `ldd /opt/<T4x>/bin64/tpitkpipe | grep not`
  - Restart the T4x processes and check if it works correctly.

- In rare cases where a logchannel is damaged so that it can no longer be opened using the Admin UI. Active Integration Gateway is checking if log channels are damaged. While defect log lines are simply skipped, T4x is not able to rescue any data if the header of the log channel is damaged. Therefore the script *Remove Logchannel* is delivered with the BGS to remove logchannels from the index and the hard drive. Follow these steps:

  - copy the affected logchannel name (e.g. "*tmp/default.log*"),
  - login BGS Admin UI and click **Script** → **Scripts** → **"Remove logchannel"**,
  - Insert the copied logchannel name in the form and run the script to delete it.

7.5 Setting Up a T4x Command Shell

A command shell with a proper Teamcenter and T4x environment is recommended to continue with T4x configuration. This is similar to the command shell that is started with the "Teamcenter Command Prompt" which is included in a standard Teamcenter installation, but with additional T4x settings.

Create a command shell with the environment variables for the desired Teamcenter database. Please edit the following example (depending on your operating system) and replace the path names with valid values for your installation.

In Windows, we recommend creating a new file *t4x_shell.bat* on your desktop for quick accessibility:

```
set TC_ROOT=C:\PLM\Teamcenter
set TC_DATA=C:\PLM\Teamcenter\tcdata
set TP_NCONHIDE=1
call %TC_DATA%\tc_profilevars
call <GS_ROOT>\etc\t4x_env.bat
cd /d %TEMP%
start "T4x command shell" cmd
```

Run this script and leave the command shell window open for further access.

The file t4x_env.bat is written during the T4x configuration in the Admin UI. So if this line causes an operating system error message, please check the configuration and the correct spelling.

The following is a T4x shell script example for UNIX (*t4x_shell.sh*):

```
TC_ROOT=/opt/teamcenter
TC_DATA=/opt/tcdata
export TC_ROOT
export TC_DATA
. $TC_DATA/tc_profilevars
. <GS_ROOT>/etc/t4x_env
```

In UNIX/Linux this script does not need the environment variable TP_NCONHIDE.

Check the settings in the command shell (same for Windows and UNIX/Linux). A good test for that is the standard Teamcenter tool `clearlocks`. If the settings are not correct, the following will not work and show errors.

Please try in your T4x command shell:

```
clearlocks –verbose
```

If the settings are correct, the output will be something like that:

```
Processes: 0, Alive: 0, Dead: 0, Remote: 0, Other: 0
```

A. T4x Workflow Handler

A.1 T4x Workflow Handler Overview

The T4x workflow handlers can be combined with all other handlers without any major modification.

Some handlers may be placed in the "Start" action of the Root Task. As Teamcenter will discard the entire workflow job if an error occurs in this action (which also means that T4x cannot store its log file Dataset), error tracking might be harder in this case so this is not recommended.

T4x Rule Handler

- **CLM4T-validate-EALogon** - Examination of the EA login.
- **T4EA-validate-EALogon** - Examination of the EA login.
- **T4O-validate-BillOfMaterial** - Check information for an Oracle EBS bill of materials transfer.
- **T4O-validate-ChangeOrder** - Validate ChangeOrder.
- **T4O-validate-EBSLogon** - Examination of the Oracle EBS login.
- **T4O-validate-GenericObject** - Check information for an Oracle EBS generic object transfer.
- **T4O-validate-Item** - Check information for an Oracle EBS item transfer.
- **T4S-check-T4SFolderExists** - Check for T4S folder.
- **T4S-validate-BillOfMaterial** - Validate information for bill of materials transfer.
- **T4S-validate-BusinessPartner** - Validate information for business partner transfer.
- **T4S-validate-ChangeNumber** - Validate ChangeNumber.
- **T4S-validate-DocumentInfoRecord** - Validate information for document info record transfer.
- **T4S-validate-DocumentStructure4BomView** - Validate information for document structure transfer.
- **T4S-validate-EquiBillOfMaterial** - Validate information for equipment bill of materials transfer.
- **T4S-validate-EquipmentMaster** - Validate information for equipment master transfer.
- **T4S-validate-FunctionalLocation** - Validate information for functional location master transfer.

- **T4S-validate-FunctionalLocationBillOfMaterial** - Validate information for functional location bill of materials transfer.
- **T4S-validate-GenericObject** - Validate information for generic object transfer.
- **T4S-validate-MaterialMaster** - Validate information for material master transfer.
- **T4S-validate-OrderBillOfMaterial** - Validate information for order bill of materials transfer.
- **T4S-validate-PIR** - Validate information for purchase info record transfer.
- **T4S-validate-SAPLogon** - Examination of the SAP login.
- **T4S-validate-Schedule** - Validate information for schedule transfer.
- **T4S-validate-Vendor** - Validate information for vendor transfer.
- **T4S-validate-WBS-BillOfMaterial** - Validate information for work breakdown structure bill of materials transfer.
- **T4X-attach-LogfileDataset** - Creates a log file for the workflow.
- **T4X-validate-GenericObject4TargetType** - Check whether all the necessary information for generic object transfer is defined.

T4x Action Handler

- **T4EA-SessionProxy** - Manage a proxy to forward the EA login data.
- **T4EA-transfer-GenericBillOfMaterial4Relation** - Generic bill of materials transfer based on a TC relation based structure.
- **T4EA-transfer-GenericObject** - Generic object transfer.
- **T4EA-transfer-GenericObject4BomLine** - Generic object transfer for each position of a bill of material.
- **T4O-SessionProxy** - Manage a proxy to forward the Oracle EBS login data.
- **T4O-define-PreferredEBSSystem** - Define the preferred Oracle EBS connection.
- **T4O-transfer-BillOfMaterial** - Bill of materials transfer to Oracle EBS.
- **T4O-transfer-CC-BOM** - CC Bill of materials transfer.
- **T4O-transfer-ChangeOrder** - Create an ECO in Oracle EBS.

- **T4O-transfer-GenericObject** - Generic object transfer.
- **T4O-transfer-Item** - Create an item in Oracle EBS.
- **T4O-transfer-Item4BomLine** - Item transfer for each position of a bill of material.
- **T4O-transfer-Routing** - Routing transfer.
- **T4S-SessionProxy** - Manages a proxy to forward the SAP login data.
- **T4S-add-Form2CC** - Add form to CC object.
- **T4S-create-Folder** - Create folder to manage workflow targets.
- **T4S-define-PreferredSapSystem** - Define the preferred SAP connection.
- **T4S-transfer-BillOfMaterial** - Bill of materials transfer to SAP.
- **T4S-transfer-BillOfMaterial4Relation** - Bill of materials transfer based on a TC relation based structure.
- **T4S-transfer-BusinessPartner** - Create or change a business partner in SAP S/4HANA.
- **T4S-transfer-CC-BOM** - CC Bill of materials transfer.
- **T4S-transfer-ChangeNumber** - Create a change number in SAP.
- **T4S-transfer-DocumentInfoRecord** - Create or change a document info record in SAP.
- **T4S-transfer-DocumentStructure4BomView** - Document structure transfer to SAP.
- **T4S-transfer-EquiBillOfMaterial** - Equipment bill of materials transfer to SAP.
- **T4S-transfer-EquipmentMaster** - Create or change an equipment master record in SAP.
- **T4S-transfer-FunctionalLocation** - Create or change a functional location master record in SAP.
- **T4S-transfer-FunctionalLocationBillOfMaterial** - Functional location bill of materials transfer to SAP.
- **T4S-transfer-GenericObject** - Generic object transfer.
- **T4S-transfer-MaterialMaster** - Create or change a material master record in SAP.
- **T4S-transfer-MaterialMaster4BomLine** - Create a material master record in SAP for every Bill of materials position.

- **T4S-transfer-OrderBillOfMaterial** - Order bill of materials transfer to SAP.
- **T4S-transfer-PIR** - Create or change a purchase info record in SAP.
- **T4S-transfer-Routing** - Routing transfer.
- **T4S-transfer-Schedule** - Create or change a schedule in SAP.
- **T4S-transfer-VariantCharacteristic** - Create a characteristic in SAP.
- **T4S-transfer-VariantClass** - Create a class of type 300 in SAP.
- **T4S-transfer-VariantValueRestriction** - Restrict characteristic values for a SAP class.
- **T4S-transfer-Vendor** - Create or change a vendor in SAP.
- **T4S-transfer-WBS-BillOfMaterial** - Bill of materials transfer to work breakdown structure bom in SAP.
- **T4S-transfer-iPPE-Node4BomLine** - Bill of materials transfer to iPPE structure node in SAP.
- **T4S-transfer-iPPE-Structure** - Bill of materials transfer to iPPE structure node in SAP.
- **T4X-attach-RevisionRule** - Attaches a dynamically created revision rule to an workflow job.
- **T4X-copy-Attributes-to-Task** - Copy attributes to task object.
- **T4X-create-AIObject4CC** - Create an Application Interface Object.
- **T4X-create-T4X-BatchJob** - This handler creates a job in the T4x BGS environment.
- **T4X-manage-Connection4Session** - Manage a proxy to store and reset preferred connections during a workflow.
- **T4X-transfer-GenericObject4TargetType** - Generic object transfer based on the specified arguments.
- **T4X-transfer-ProductConfigurator** - Transfer product configurator objects to an external system.

A.2 T4x Workflow Rule Handler Documentation

- **CLM4T-validate-EALogon**
  This handler validates the EA login to SIMATIC IT.
  The handler should be used if possible in the "Complete" action of the SIT logon task.

  -use_EA_system
  If set, CLM4T will use the EA system stated as the value of this argument.

- - -

- **T4EA-validate-EALogon**
  This handler validates the EA login.
  The handler should be used if possible in the "Complete" action of the EA logon task.

  -use_EA_system
  If set, T4EA will use the EA system stated as the value of this argument.

  - - -

- **T4O-validate-BillOfMaterial**
  This rule handler checks whether all the necessary information for an Oracle EBS bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions "Perform" and "Complete" (equivalent to the argument -Check=MANDATORYFIELDS) return valid values. All other actions return "EPM_undecided".
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]

or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4O-validate-ChangeOrder**
  This rule handler checks if a change number was given and may additionally check if an ECO with this Name exists in Oracle EBS.
  The rule handler may be placed in the "Start" action of the Root Task. Then it will not be executed for every target, but only once in the Wokflow job. It also has the advantage that T4O may change the Workflow Job Name; later Teamcenter will not allow that.
  If the argument "-check_EPM_Targets=True" is set, it does not check the Job Name as ECO Name, but you can set it freely in the mapping and differently for every target of this Workflow job. If this is not set, the handler will only read the Workflow Job Name and use this as ECO Name, no matter of the number of targets. But nevertheless, the mapping needs to point to this information:
  set ::EBSDat(ChangeOrder:P_ECO_REC:ECO_NAME) [tpco_formatHEX16 $::TcData(Workflow:JobName)]

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4O-validate-EBSLogon**
  This handler validates the Oracle EBS login.
  The handler should be used if possible in the "Complete" action of the EBS logon task.

  -use_EBS_system
  If set, T4O will use the Oracle EBS system stated as the value of this argument.

  - - -

- **T4O-validate-GenericObject**
  This rule handler checks whether all the necessary information for an Oracle EBS generic object transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions "Perform" and "Complete" (equivalent to the argument -Check=MANDATORYFIELDS) return valid values. All other actions return "EPM_undecided".
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule

Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4O-validate-Item**
  This rule handler checks whether all the necessary information for an Oracle EBS item transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions "Perform" (equivalent to the argument -Check= MANDATORYFIELDS) and "Complete" (equivalent to the argument -Check=ITEMNUMBER) return valid values. All other actions return "EPM_undecided".
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]

Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-check-T4SFolderExists**
  This handler checks whether the T4S folder (in order to manage workflow targets) exists.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties

to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-BillOfMaterial**
  This handler checks whether all the necessary information for a bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” and "Complete“ (equivalent to the argument "-check=MANDATORYFIELDS") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
  Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
  In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
  (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-BusinessPartner**
  This handler checks whether all the necessary information for a business partner transaction is defined.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects, e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with

T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status
Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision"
"object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info"
"T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification
[::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-ChangeNumber**
  This handler checks if a change number was given and may additionally check if an ECM with this ID exists in SAP.

The rule handler may be placed in the "Start" action of the Root Task. Then it will not be executed for every target, but only once in the Wokflow job. It also has the advantage that T4S may change the Workflow Job Name; later Teamcenter will not allow that.

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-DocumentInfoRecord**
  This handler checks whether all the necessary information is defined for the creation of a document info record or if the reverse mapping of the SAP document key was successful, respectively.
  The handler can be used in any task, preferably in action "Complete".

  -use_related_objects
  If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer. This can be used to create SAP document structures based on relations.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties

to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-DocumentStructure4BomView**
  This handler checks whether all the necessary information for a document structure transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” and "Complete“ (equivalent to the argument "-check=MANDATORYFIELDS") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
  Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
  In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
  (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-EquiBillOfMaterial**
  This handler checks whether all the necessary information for an equipment bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” and "Complete“ (equivalent to the argument "-check=MANDATORYFIELDS") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].

The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status
Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision"
"object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info"
"T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification
[::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-EquipmentMaster**

This handler checks whether all the necessary information for a equipment master transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler.
The handler can be used in any task, preferably in action "Complete".

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-FunctionalLocation**
  This handler checks whether all the necessary information for a functional location master transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects, e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with

T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status
Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision"
"object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info"
"T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification
[::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-FunctionalLocationBillOfMaterial**
  This handler checks whether all the necessary information for an functional location bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” and "Complete“ (equivalent to the argument "-

check=MANDATORYFIELDS") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-GenericObject**
  This handler checks whether all the necessary information for a generic object transaction is defined. The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,

e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-MaterialMaster**
  This handler checks whether all the necessary information for a material master transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” (equivalent to the argument "-check=MANDATORYFIELDS") and "Complete“ (equivalent to the argument "-check=MATERIALNUMBER") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.

Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-OrderBillOfMaterial**
  This handler checks whether all the necessary information for an order bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler. Only the actions “Perform” and "Complete“ (equivalent to the argument "-check=MANDATORYFIELDS") return valid values. All other actions return "EPM_undecided". The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]

or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-PIR**
  This handler checks whether all the necessary information for a purchase info record transaction is defined.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.

Specify e.g. -
AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode

This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-SAPLogon**
  This handler validates the SAP login.
  The handler should be used if possible in the "Complete" action of the SAP logon task.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

- **T4S-validate-Schedule**
  This handler checks whether all the necessary information for a schedule transaction is defined.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties

to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-Vendor**
  This handler checks whether all the necessary information for a vendor transaction is defined.
  The handler can be used in any task, preferably in action "Complete".

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,
  e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
  T4X_AddObject4MappingPref =
  EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
  e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
  or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
  Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
  In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
  (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

  -TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4S-validate-WBS-BillOfMaterial**
  This handler checks whether all the necessary information for a work breakdown structure bill of materials transaction is defined. If no arguments are transferred, the action is used as a parameter for the function of the rule handler.
  The handler can be used in any task, preferably in action "Complete".

  -use_VariantRuleTransfer
  If set to TRUE, the VariantRule information is considered during BOM transfer.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

  -bomview_prio_list
  This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition

If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]

or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

- **T4X-attach-LogfileDataset**
  This rule handler creates an additional workflow log file that will be linked to the workflow as an attachment. In case of an error, the error message defined with storeMessage2 will be shown there. Additionally, for writing into this file no matter if an error occurred, use the function storeMessage2DatasetLogfile instead, e.g. ::T4X::CORE::storeMessage2DatasetLogfile "MAPPING" "Test message".
  The rule handler should be used if possible in the "Start" action of the Root Task so that all subsequent handlers can use the log file.
  During a Workflow Job run, the log file is created in the "Attachments" of this Workflow Job (check in "Workflow Viewer").

Note:

If this handler is used, the log information that is written into this file will not be written to the session log file any more.

-acl
This parameter defines the Access Control List (acl) name that is attached to the log file dataset, so that it is possible to update the dataset by different users.

- - -

- **T4X-validate-GenericObject4TargetType**
  No mandatory arguments, all handler arguments are optional.

  -BomHeaderTypeList
  Required for BOM transfers. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

  -ChangeType
  The parameter defines the preference prefix ([ProductKey][ChangeType]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

  -ChangeTypeName
  The parameter defines the preference prefix ([ProductKey][ChangeTypeName]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

  -ProductKey
  The parameter specifies the T4x product, e.g. T4S.

  -use_EA_system
  If set, T4X will use the EA system stated as the value of this argument.

  -use_EBS_system
  If set, T4O will use the Oracle EBS system stated as the value of this argument.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

-useView4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on BOM view information and the TCL namespace for the mapping switches from GENOBJ to GENBOM.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]

(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-check
This parameter is used in the combination with the -user_rule=Default argument to trigger target object specific build in rules.

-mode
This parameter is used in the combination with the -user_rule=Default and the -check argument to trigger target object specific build in rules.

-user_rule
Name of the rule to check if the mapping was executed (default value is "Default"). If set to Default, a target object specific check is executed for the mandatory input fields. Otherwise a corresponding mapping function (callCustomerRuleHandler) is called to calculate the rule handler decision.

- - -

A.3 T4x Workflow Action Handler Documentation

- **T4EA-SessionProxy**
  This handler manages a proxy to forward the EA login data between different TC users if they are involved in one workflow, i.e. only the first user will have to type his or her EA login data and every EA transaction in the same Workflow job will be done with the same EA user. Add "-Mode=SAVE" after the EA Logon Task. If another TC user gets the workflow (e.g. a reviewer), add "-Mode=READ" in his task to enable him sending data to EA with the same EA user as the TC user who started the workflow without the need to login separately. As then, this TC user automatically uses the same EA login outside of any Workflow, too, you should add "-Mode=FORGET" to avoid that. This can also be used to restore the previous preferred connection.
  The handler can be used in any task, preferably in action "Complete".

Note:

The EA logon data must be present and correct. This is best checked with the "T4EA-validate-EALogon" rule handler before the T4EA-SessionProxy called with the -Mode=SAVE argument.

Do not use this handler with "-Mode=FORGET" in a Workflow Task that is executed before the Workflow is shifted to another Teamcenter user (e.g. a generic T4x job user), because this would delete the current user's EA login (for Workflow and interactive use as well).

-Mode
Possible values are:
SAVE - saves the current EA login into a T4EA proxy.
READ - reads the EA login from a T4EA proxy.
FORGET - deletes the logon data from the T4EA proxy without deleting the T4EA Proxy itself.
DELETE - deletes a T4EA proxy. (*)
(*) The difference is that after FORGET the login data are not used automatically again, but they may be reused with the argument -Mode=READ; after a DELETE, they cannot be restored.

- - -

- **T4EA-transfer-GenericBillOfMaterial4Relation**
  This handler creates or changes a Bill of materials based on a TC relation based structure (instead of using a TC bom view).

  -add_related_objects
  If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

  -bom4rel_use_bomview
  Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

  -Bom4RelationTargetTypeName
  The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

  -BomHeaderTypeList
  The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

  -use_EA_system
  If set, T4EA will use the EA system stated as the value of this argument.

  - - -

  -bom_rev_rule

This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].

The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status
Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision"
"object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info"
"T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification
[::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4EA-transfer-GenericObject**
  For an element of object type ItemRevision a generic object transfer to an Enterprise Application is executed. A configured Teamcenter object is transferred via a Teamcenter workflow to a specified EA target object. In the Enterprise Application, the defined object is created or updated. The business logic for the creation or update of the EA object is defined and implemented by the customer in the mapping file. Supported Objects: ItemRevision based Objects (included attached Forms and datasets) Bom View Revisions. Define the generic object mapping for a specific type in your own mapping file in the namespace "::T4EA::GENOBJ::CUSTOM::MAPPING" (do not forget to source it in t4ea_mapping_config.sd) in the following functions:
  TC_Object2EA_Object – for defining the data mapping from Teamcenter to the EA object
  EA_Object2TC_Object – for defining the back mapping from the EA object to Teamcenter
  performTransfer – for customer specific definition of the actual transfer to the Enterprise Application
  getObjectInfo – for retrieving the object information from the EA object

  -add_related_objects
  If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

  -bom4rel_use_bomview
  Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

  -Bom4RelationTargetTypeName
  Required for BOM transfers based on relation information. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

  -BomHeaderTypeList

Required for BOM transfers. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-ChangeTypeName
The parameter defines the preference prefix ([ProductKey][ChangeTypeName]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

-use_EA_system
If set, T4EA will use the EA system stated as the value of this argument.

-use_EPM_objects_as_targets
With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task and current_task. Instead it is also possible to use the EPM_target_attachment (default) In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

-useBom4Relation4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on relation information.

-useView4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on BOM view information and the TCL namespace for the mapping switches from GENOBJ to GENBOM.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention:

[ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set:
[ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]

Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix

This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4EA-transfer-GenericObject4BomLine**
  This handler transfers from the currently selected Teamcenter BOM every position object as well as the BOM header object to an Enterprise Application. So if it is requested then this additional handler should be used before the handler T4EA-transfer-GenericObject configured for BOM transfer.

  -use_EA_system
  If set, T4EA will use the EA system stated as the value of this argument.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

  -bomview_prio_list
  This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

  -hide_GDELines
  If set to TRUE, T4x will hide GDE BOM positions.

  -hide_incremental_changes_bomlines
  If set to TRUE, T4x will hide incremental changes.

  -hide_substitutes
  If set to TRUE, T4x will hide substitutes.

  -hide_suppressed_bomlines
  If set to TRUE, T4x will hide suppressed BOM positions.

  -hide_unconfigured_bomlines
  If set to TRUE, T4x will hide unconfigured variants.

  -hide_variants_bomlines
  If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping

This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode

If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-SessionProxy**
  This handler manages a proxy to forward the Oracle EBS login data between different TC users if they are involved in one workflow, i.e. only the first user will have to type his or her Oracle EBS login data and every Oracle EBS transaction in the same Workflow job will be done with the same Oracle EBS user. Add "-Mode=SAVE" after the Oracle EBS Logon Task. If another TC user gets the workflow (e.g. a reviewer), add "-Mode=READ" in his task to enable him sending data to Oracle EBS with the same Oracle EBS user as the TC user who started the workflow without the need to login separately. As then, this TC user automatically uses the same Oracle EBS login outside of any Workflow, too, you should add "-Mode=FORGET" to avoid that. This can also be used to restore the previous preferred connection.
  The handler can be used in any task, preferably in action "Complete".

> Note:
>
> The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler before the T4O_SessionProxy-AH called with the "-Mode=Save" argument.
>
> Do not use this handler with "-Mode=Forget" in a Workflow Task that is executed before the Workflow is shifted to another Teamcenter user (e.g. a generic T4O job user), because this would delete the current user's EBS login (for Workflow and interactive use as well).

-Mode
Possible values are:
SAVE saves the current Oracle EBS login into a T4O proxy
READ reads the Oracle EBS login from a T4O proxy
FORGET deletes the logon data from the T4O proxy without deleting the T4O Proxy itself
DELETE deletes a T4O proxy (*)
(*) The difference is that after FORGET the login data are not used automatically again, but they may be reused with the argument -Mode=READ; after a DELETE, they cannot be restored.

- - -

- **T4O-define-PreferredEBSSystem**
  This handler defines the preferred Oracle EBS connection for all following transfer handlers.
  The handler can be used in any task, preferably in action "Complete".

> Note:
>
> The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

- - -

- **T4O-transfer-BillOfMaterial**
  This handler transfers for each configured BOM view type a defined bill of materials to Oracle EBS.
  The handler can be used in any task, preferably in action "Complete".

> Note:
>
> The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

-LOOP_COUNTER
Optional for BOM Rule Imports. This parameter specifies how often T4O should try to get the rule import result.

-LOOP_DELAY
Optional for BOM Rule Imports. This parameter specifies the time between the attempts to get the rule import result.

-RETRIEVE_IMPORT_RESULT
Optional for BOM Rule Imports. If set to TRUE, polling for import result is enabled. As the BOM Rule Import will execute an EBS concurrent program which is asynchronous, T4O will try to get the import result every x minutes. We recommend to set this argument only for use in test environments.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

-use_VariantRuleTransfer
If set to TRUE, the VariantRule information is considered during BOM transfer.

-VARIANT_TRANSFER
Required for BOM Rule Imports. If set to TRUE, the BOM Rule Import to Oracle Configurator is enabled.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines

If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration

should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-CC-BOM**
  Transfers MBOM Data extracted from the Application Interface Object found as target attachment of the current task to Oracle EBS.

The handler can be used in any task, preferably in action "Complete".

> Note:
>
> Works only for objects of type Application Interface Object.
>
> The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]

Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix

This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-ChangeOrder**
  This handler creates an ECO in Oracle EBS on the basis of the workflow job name.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

  -check_EPM_Targets
  If set to TRUE, the handler reads data from each Workflow job target data additionally.

  -UPDATE_EXISTING
  If set to FALSE, the ECO is created but not modified. The handler checks first if the ECO already exists and if it does, this handler aborts the transaction.

  -use_EBS_system
  If set, T4O will use the Oracle EBS system stated as the value of this argument.

  -use_EPM_objects_as_targets
  With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task (Default) and current_task.
  In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties

to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-GenericObject**
  A configured Teamcenter object is transferred via a Teamcenter workflow to a specified EBS target object. In Oracle, the defined object is created or updated and depending on the business rules approved or rejected. The handler can be used in any task, preferably in action "Complete".
  The business logic for the creation or update of the EBS object is defined and implemented by the customer in the corresponding mapping file.
  Supported Objects: all Teamcenter POM object types and BOM Views in the case of a Structure based transfer. Note that for Generic Object handlers, the same concept of preferences for data extraction is used as for all out of the box object types. Also the data mapping for Generic Object handlers is done in the same way as for out of the box object types: Define the Generic Object mapping for a specific type in your own mapping file (don't forget to source it in t4o_mapping_config.sd) in the following functions:
  TC_Object2EBS_Object – for defining the data mapping from Teamcenter to Oracle EBS.
  EBS_Object2TC_Object – for defining the back mapping from Oracle EBS to Teamcenter.
  performTransfer – for customer specific definition of the actual transfer to EBS.
  getObjectInfo – for retirieving the object information from EBS.
  callCustomerRuleHandler – for defining the customer specific rule handler.
  TC_Object2EBS_ObjectPosition – only needed for BOM views; for defining the BOM position mapping from Teamcenter to Oracle EBS

Note:

The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

-add_related_objects
If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

-bom4rel_use_bomview
Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

-Bom4RelationTargetTypeName
Required for BOM transfers based on relation information. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-BomHeaderTypeList
Required for BOM transfers. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-ChangeTypeName
The parameter defines the preference prefix ([ProductKey][ChangeTypeName]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

-use_EPM_objects_as_targets
With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task and current_task. Instead it is also possible to use the EPM_target_attachment (default) In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

-useBom4Relation4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on relation information.

-useView4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on BOM view information and the TCL namespace for the mapping switches from GENOBJ to GENBOM.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention:
[ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention:
[ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set:
[ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision

or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error

If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-Item**
  This handler creates or alters an item in Oracle EBS.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

  -MODEL_BOM
  If set to TRUE, the EBS item attribute "Bom Item Type" is set to "Model" instead of "Standard" (as by default).

  -use_EBS_system
  If set, T4O will use the Oracle EBS system stated as the value of this argument.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.

Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-Item4BomLine**
  This handler transfers from the currently selected Teamcenter BOM every position object as well as the BOM header object as Inventory item to Oracle EBS. This may be helpful because the EBS does not allow creating/updating a BOM if no items for the header and the components have been created yet. If required, this additional handler should be used before the standard T4O BOM handler T4O-transfer-BillOfMaterial.

  > Note:
  >
  > Same as the standard T4O BOM processing, this handler does not do anything with sub BOM levels but only with the objects in the current BOM level, i.e. the selected BOM header and its direct BOM positions.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level

This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4O-transfer-Routing**
  Transfers Routing Data extracted from the Application Interface Object found as target attachment of the current task to Oracle EBS. After the creation of the routing in Oracle EBS, group counter and group id information is written back into TC in the form attached to the CC Object contained in the Application Interface Object.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > Works only for objects of type Application Interface Object. A form with at least two attributes (for group counter and group id) must be attached to the CC Object contained in the Application Interface Object.
  >
  > The EBS logon data must be present and correct. This is best checked with the "T4O-validate-EBSLogon" rule handler.

  -use_EBS_system
  If set, T4O will use the Oracle EBS system stated as the value of this argument.

  - - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode

If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-SessionProxy**
  This handler manages a proxy to forward the SAP login data between different TC users if they are involved in one workflow, i.e. only the first user will have to type his or her SAP login data and every SAP transaction in the same Workflow job will be done with the same SAP user. Add "-Mode=Save" after the SAP Logon Task. If another TC user gets the workflow (e.g. a reviewer), add "-Mode=READ" in his task to enable him sending data to SAP with the same SAP user as the TC user who started the workflow without the need to login separately. As then, this TC user automatically uses the same SAP login outside of any Workflow, too, you should add "-Mode=FORGET" to avoid that. This can also be used to restore the previous preferred connection.
  The handler can be used in any task, preferably in action "Complete".

> Note:
>
> The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler before the T4S-SessionProxy called with the -Mode=Save argument.
>
> Do not use this handler with "-Mode=Forget" in a Workflow Task that is executed before the Workflow is shifted to another Teamcenter user (e.g. a generic T4S job user), because this would delete the current user's SAP login (for Workflow and interactive use as well).

-Mode
Possible values are:
SAVE saves the current SAP login into a T4S proxy
READ reads the SAP login from a T4S proxy
FORGET deletes the logon data from the T4S proxy without deleting the T4S Proxy itself
DELETE deletes a T4S proxy (*)
(*) The difference is that after FORGET the login data are not used automatically again, but they may be reused with the argument -Mode=READ; after a DELETE, they cannot be restored.

- - -

- **T4S-add-Form2CC**
  This handler adds the form specified by handler argument FORM_TYPE to the CC object found as target attachment of current task. If handler argument ADD_AS_TARGET is also specified, the form will be added also as target attachment of the current task.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > Works only for objects of type CC object.
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -ADD_AS_TARGET
  If set to TRUE, the newly created form is added as a workflow target attachment. Otherwise the form is only attached to the CC.

  -FORM_TYPE
  Mandatory. The parameter specifies form type name of form to be added

  -RESET_FORM
  If set to TRUE, the content of the dorm is reset.

  - - -

  -collect_all_errors
  If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-create-Folder**
  This handler creates a folder in order to manage workflow targets. The reason is that it adds all workflow targets into that new created workflow folder and deletes all that finished OK, so in case of an error the current target is not deleted from that folder so it can be checked easily which target had an error: those that are still in that folder.
  The handler can be used in any task, preferably in action "Complete".

  -sap_object_type
  This parameter specifies the object types. Possible values are: MM, DIR, BOM, ECN (in different possible combinations, colon separated), or "all"

  - - -

  -collect_all_errors
  If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

  -continue_on_error

If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-define-PreferredSapSystem**
  This handler defines the preferred SAP connection for all following transfer handlers, i.e. as long as no other SAP connection is given, this one will be used.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

- **T4S-transfer-BillOfMaterial**
  This handler transfers for each configured BOM view type a defined bill of materials to SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  -use_VariantRuleTransfer
  If set to TRUE, the VariantRule information is considered during BOM transfer.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

  -bomview_prio_list
  This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition

If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]

or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-BillOfMaterial4Relation**
  This handler creates or changes a SAP bill of materials based on a TC relation based structure (instead of using a TC bom view).
  The handler can be used in any task, preferably in action "Complete".

  -add_related_objects
  If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

  -bom4rel_use_bomview
  Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision

or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error

If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-BusinessPartner**
  This handler creates or changes a business partner in SAP S/4HANA system.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-CC-BOM**
  Transfers MBOM Data extracted from the Application Interface Object found as target attachment of the current task to SAP.

The handler can be used in any task, preferably in action "Complete".

> Note:
>
> Works only for objects of type Application Interface Object.
>
> The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties

In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result

This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-ChangeNumber**
  This handler creates a change number in SAP on the basis of the workflow job name.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -check_EPM_Targets
  If set to TRUE, the handler reads data from each Workflow job target data additionally.

  -UPDATE_EXISTING
  If set to FALSE, the ECM is created but not modified. The handler checks first if the ECM already exists and if it does, this handler aborts the transaction.

  -use_EPM_objects_as_targets
  With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task (Default) and current_task.
  In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-DocumentInfoRecord**
  This handler creates a new document info record in SAP for a given data set with all the metadata (including the original files).

The handler can be used in any task, preferably in action "Complete".

> Note:
>
> The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

-documentstructure
Possible values are:
YES - create/update the document structure of the DIR.
NO (default) - do not update the document structure of the DIR.

-metadata
Possible values are:
YES (default) - create/update the metadata of the DIR.
NO - do not update the metadata of the DIR.

-mode
Possible values are:
CREATE - the DIR is only created, not updated (if the DIR already exists the workflow stops with an error).
UPDATE (default) - create the DIR or update it if it exists.

-objectlinks
Possible values are:
YES (default) - create/update the object links of the DIR.
NO - do not update the object links of the DIR.

-ORIGINALS
Possible values are:
YES (default) - create/update the original data (i.e. the files) of the DIR.
NO - do not update the original data (i.e. the files) of the DIR.

-process_target_dataset
If set to TRUE, T4S processes a Dataset as a workflow target (by default it uses an Item Revision only). Note that only Dataset types are used which are in the preference T4S_DocumentInfoRecordTypeList and it only works correctly if that Dataset is stored in an Item type with the references defined in the DIR mapping preferences for that Dataset type.

-use_related_objects
If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer. This can be used to create SAP document structures based on relations.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system

If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]

(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-DocumentStructure4BomView**
  This handler transfers for each configured BOM view type a defined document structure to SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  -use_VariantRuleTransfer
  If set to TRUE, the VariantRule information is considered during BOM transfer.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

  -bomview_prio_list
  This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

  -hide_GDELines
  If set to TRUE, T4x will hide GDE BOM positions.

  -hide_incremental_changes_bomlines
  If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule

If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set:
[ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.

In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-EquiBillOfMaterial**
  This handler transfers for each configured BOM view type a defined equipment bill of materials to SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  -use_VariantRuleTransfer
  If set to TRUE, the VariantRule information is considered during BOM transfer.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

  -bomview_prio_list
  This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

  -hide_GDELines
  If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention:

[ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set:
[ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]

Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification] (Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix

This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-EquipmentMaster**
  This handler creates or changes an equipment master record in SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
  to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
  or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
  to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
  In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
  The preference [PreferenceName] contains the definition for all needed objects,

e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status
Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision"
"object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping
"RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info"
"T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification
[::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result

This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-FunctionalLocation**
  This handler creates or changes a functional location master record in SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]

For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-FunctionalLocationBillOfMaterial**
  This handler transfers for each configured BOM view type a defined functional location bill of materials to SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

-use_VariantRuleTransfer
If set to TRUE, the VariantRule information is considered during BOM transfer.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping

This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode

If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-GenericObject**
  A configured Teamcenter object is transferred via a Teamcenter workflow to a specified SAP target object. In SAP, the defined object is created or updated and depending on the business rules approved or rejected. The business logic for the creation or update of the SAP object is defined and implemented by the customer in the corresponding mapping file.
  Supported Objects: all Teamcenter POM object types and BOM Views in the case of a Structure based transfer. Note that for Generic Object handlers, the same concept of preferences for data extraction is used as for all other out of the box object types.
  Also the data mapping for Generic Object handlers is done in the same way as for out of the box object types: Define the Generic Object mapping for a specific type in your own mapping file (don not forget to source it in t4s_mapping_config.sd) in the following functions:

TC_Object2SAP_Object – for defining the data mapping from Teamcenter to SAP
TC_Object2SAP_ObjectPosition – for defining the data mapping from Teamcenter BOM Position to SAP. Only for BOM views.
SAP_Object2TC_Object – for defining the back mapping from SAP to Teamcenter
performSapTransfer – for customer specific definition of the actual transfer to SAP
getObjectInfo – for retirieving the object information from SAP
callCustomerRuleHandler – for defining a customer specific rule handler

> Note:
>
> The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

-add_related_objects
If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

-bom4rel_use_bomview
Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

-Bom4RelationTargetTypeName
Required for BOM transfers based on relation information. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-BomHeaderTypeList
Required for BOM transfers. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-ChangeTypeName
The parameter defines the preference prefix ([ProductKey][ChangeTypeName]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

-use_EPM_objects_as_targets
With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task and current_task. Instead it is also possible to use the EPM_target_attachment (default) In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

-useBom4Relation4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on relation information.

-useView4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on BOM view information and the TCL namespace for the mapping switches from GENOBJ to GENBOM.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines

If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find

the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-MaterialMaster**
  This handler creates or changes a material master record in SAP.
  The handler can be used in any task, preferably in action "Complete".

Note:

The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,

e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result

This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-MaterialMaster4BomLine**
  This handler transfers from the currently selected Teamcenter BOM every position object as well as the BOM header object as Material Master to SAP. This may be helpful because SAP does not allow creating/updating a BOM with a position that is not a SAP MM yet. So if it is requested then this additional handler should be used before the standard T4S BOM handler.

  > Note:
  >
  > Same as the standard T4S BOM processing, this handler does not do anything with sub BOM levels
  >
  > but only with the objects in the current BOM level, i.e. the selected BOM header and its direct BOM positions.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref = EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-OrderBillOfMaterial**
  This handler transfers for each configured BOM view type a defined order bill of materials to SAP. The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  -use_VariantRuleTransfer
  If set to TRUE, the VariantRule information is considered during BOM transfer.

  - - -

  -bom_rev_rule
  This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref = EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-PIR**
  This handler creates or changes a purchase info record in SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-Routing**
  Transfers Routing Data extracted from the Application Interface Object found as target attachment of the current task to SAP. After the creation of the routing in SAP, group counter and group id information is written back into TC in the form attached to the CC Object contained in the Application Interface Object.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > Works only for objects of type Application Interface Object. A form with at least two attributes (for group counter and group id) must be attached to the CC Object contained in the Application Interface Object.
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -Target
  This parameter is used to select the specific target routing type in SAP. Possible values are:
  Routing (default)
  InspectionPlan

RefSetOfOperations

-use_ai_as_target
If set to TRUE, the AI object based PLMXML transfer is enabled instead of the routing net change functionality. Currently only used in the context of the T4S routing transfer.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,

e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result

This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-Schedule**
  This handler creates or changes a schedule in SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
  As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

  - - -

  -AddObject4Mapping
  This parameter allows T4x reading additional Teamcenter data during the transaction.
  Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
  For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
  Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision

to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-VariantCharacteristic**
  This handler creates a characteristic in SAP based on an OptionFamily in context of Teamcenter Product Configurator.

  -revision_rule
  This parameter specifies the revision rule name that should be applied to the product configuratior context during the transfer.

  -revision_rule_preference
  This parameter specifies the name of a preference that configures the revision rule name that should be applied to the product configuratior context during the transfer. This approach allows a revision rule name change without changing the workflow template.

  -revision_rule_relation
  This parameter specifies the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer.

  -revision_rule_relation_preference

This parameter specifies the name of a preference that configures the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer. This approach allows to change the relation to the revisions rule without changing the workflow template.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,

e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result

This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-VariantClass**
  This handler creates a class of type 300 in SAP based on a Dictionary or a Configurator context in context of Teamcenter Product Configurator. The characteristics which correspond to Teamcenter OptionFamilies are linked to the class.

  -revision_rule
  This parameter specifies the revision rule name that should be applied to the product configuratior context during the transfer.

  -revision_rule_preference
  This parameter specifies the name of a preference that configures the revision rule name that should be applied to the product configuratior context during the transfer. This approach allows a revision rule name change without changing the workflow template.

  -revision_rule_relation
  This parameter specifies the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer.

  -revision_rule_relation_preference
  This parameter specifies the name of a preference that configures the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer. This approach allows to change the relation to the revisions rule without changing the workflow template.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-VariantValueRestriction**
  This handler restricts characteristic values for a SAP class in context of Teamcenter Product Configurator. If in Teamcenter a Product Context Option contains less values than the Dictionary

Option this will be represented in SAP by a Constraint of type restriction and a Variant Table which limits the available values.

-revision_rule
This parameter specifies the revision rule name that should be applied to the product configuratior context during the transfer.

-revision_rule_preference
This parameter specifies the name of a preference that configures the revision rule name that should be applied to the product configuratior context during the transfer. This approach allows a revision rule name change without changing the workflow template.

-revision_rule_relation
This parameter specifies the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer.

-revision_rule_relation_preference
This parameter specifies the name of a preference that configures the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer. This approach allows to change the relation to the revisions rule without changing the workflow template.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).

Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error

If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-Vendor**
  This handler creates or changes a vendor in SAP.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > The SAP logon data must be present and correct. This is best checked with the "T4S-validate-SAPLogon" rule handler.

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system
  If set, T4S will use the SAP system stated as the value of this argument.
  If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
  E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.

As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-WBS-BillOfMaterial**
  This handler transfers for each configured BOM view type a defined work breakdown structure bill of materials to SAP.

The handler can be used in any task, preferably in action "Complete".

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

-use_VariantRuleTransfer
If set to TRUE, the VariantRule information is considered during BOM transfer.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find

the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName

The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-iPPE-Node4BomLine**
  This handler transfers all BOM lines to the corresponding iPPE structure node in SAP, so that the selection conditions and the assigned material can be updated in SAP.

The handler can be used in any task, preferably in action "Complete".

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4S-transfer-iPPE-Structure**
  This handler transfers all BOM lines to the corresponding iPPE structure node in SAP, so that the selection conditions and the assigned material can be updated in SAP.
  The handler can be used in any task, preferably in action "Complete".

  -use_SAP_client
  If set, T4S will use the SAP client stated as the value of this argument.

  -use_SAP_system

If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if

a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]

For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -
AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4X-attach-RevisionRule**
  The handler can be used in any task, preferably in action "Start" of the perform-signoff task. It is also possible to use the handler in a DO-task as well. This handler creates a dynamic revision rule and attaches it to the current workflow job for telling another T4x handler later in the same workflow job how to find the desired data from Teamcenter. This allows T4x handling the Teamcenter BOM occurrence effectivity in a more flexible way than the standard fix defined revision rules
  The "last_release_status" is a real Teamcenter release status and this value determines if T4x should use the date, the unit or the end_item of that Teamcenter status object. In contrast to that, the "EPM_release_status_attachment" is a temporary status object attached by T4x; the effectivity determination by the status date, unit or the end_item of that status object works the same. If no argument is specified the handler returns with an error message. The handler can be used in any task, preferably in action "Complete". Example how to use this handler for a T4S BOM workflow in order to use the dynamic revision rule "MyRevRule": set three handlers in the "Complete" action of a T4S BOM workflow task with the handler arguments as stated here: That means a new revision rule is created

based on the revision rule template "MyRevRule", it is attached to the workflow target and used by the T4S BOM handler. The third handler will remove the revision rule then If the argument "-input" has last_release_status set as the first part of the string, the data will be retrieved from the effectivity of the status set in the last_release_status attribute. If the first part of the argument "-input" value is EPM_release_status_attachment, the data will be retrieved from the effectivity of the status created in the workflow and attached as EPM_release_status_attachment. The second part of the "-input" value describes type of data which shall be read and used:
date stands for the effectivity date
unit stands for the effectivity unit
end_item stands for the effectivity end_item

> Note:
>
> As for now for retrieving the data the first found effectivity and date will be used!

-mode
Possible values are:
ADD (default) will attach the specified revision rule to current workflow target.
DELETE will remove the attached revision rule from the workflow target and delete it from the data base.

-rev_rule_relation
This parameter specifies the relation name or reference attribute that is used to attach the revision rule to the workflow job (required).

-use_date
If set to TRUE, the date effectifty information is copied from the ReleaseStatus object found via the -use_status_relation argument to the temporary Revision Rule.

-use_end_item
If set to TRUE, the end item effectifty information is copied from the ReleaseStatus object found via the -use_status_relation argument to the temporary Revision Rule.

-use_rev_rule
This parameter specifies the revision rule template name that is required by the -mode=add argument.

-use_status_relation
This parameter specifies the relation to the Teamcenter ReleaseStatus object that is used as input for the temporary Revision Rule.

-use_unit
If set to TRUE, the unit effectifty information is copied from the ReleaseStatus object found via the -use_status_relation argument to the temporary Revision Rule.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4X-copy-Attributes-to-Task**
  This handler copies the given literal value to the given attribute of the task object. In T4EA this handler is used to pass static values to the GUI, especially the EA connection identifier. The handler is not restricted to T4EA, however. The handler can be used in any action, preferably in action "Start". Every argument name is interpreted as the internal name of an attribute of the EPM task object, this handler is attached to.
  Example: the argument "-eax2TargetTypeName=Article" will cause the handler to write the value "Article" to the attribute named "eax2TargetTypeName" of the EPMTask. This attribute only exists on a specific derivation of EPMTask on the type EAX2LogonTask. With this functionality it is possible to configure a workflow to choose an EA connection that is compatible with the given TargetTypeName. The GUI will restrict the logon dialog to the corresponding connections.
  Here is a type-independent example: "-object_name=My special task" will set the task name to "My special task" at runtime.

  -collect_all_errors

If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4X-create-AIObject4CC**
  This handler creates an Application Interface Object of type specified in handler argument for the CC Object found as target attachment of the current task.
  The handler can be used in any task, preferably in action "Complete".

  > Note:
  >
  > Works only for objects of type CC object.

  -AI_TYPE
  Defines the resulting Application Interface Object Type Name that should be created based on the profived target objects. Example:
  -AI_TYPE=EBS2AI4Routing Mandatory: Application Interface Object Type Name

  -ProductFlavor

Stores the string to the job attribute JobFilter which can be used to control which job agent may execute the job.

-TargetTypeName
Stores the string to the job attribute JobFilter which can be used to control which job agent may execute the job.

-use_cc_name
If set to TRUE, the name of the newly created AI object is based on the CC object that is used as an input for the AI object. We recommend setting this to true.

-use_export_transfer_mode
Name of the PLMXML export transfer mode. The parameter overwrites the defined default name of the PLMXML export transfer mode for the Application Interface Object Type. The PLMXML export transfer type is defined in Teamcenter internally in the PLMXML application.

-use_import_transfer_mode
Name of the PLMXML import transfer mode. The parameter overwrites the defined default name of the PLMXML import transfer mode for the Application Interface Object Type. The PLMXML import transfer type is defined in Teamcenter internally in the PLMXML application.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference

With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4X-create-T4X-BatchJob**
  No mandatory arguments, all handler arguments are optional.
  The handler can be used in any task, preferably in action "Start" of the perform-signoff task. It is also possible to use the handler in a DO-task as well.

  -Filter
  Stores the string to the job attribute JobFilter which can be used to control which job agent may execute the job.

  -JobNamePrefix
  This will manipulate the job name in the job queue. Use this to be able to differentiate by job name which workflow created the job.
  Valid Keywords:
  $SITE
  $USERID
  $USERNAME
  $GROUP
  $ROLE

  -JobPrio
  Integer argument assigns a priority specifically to the workflow job different from the default value 50. A lower value signifies a higher priority. Jobs with high priority are executed first.

  -JobTimeout
  Assigns the timeout value in seconds for the job. If a job execution is not finished in the time frame the job will abort with status RUNTIME ERROR.

  -tc_user
  Sets the job attribute ITK_TC_USER_ID to a specific TC user to use for the ITK connection of the job. The following values are supported:
  SelectedReviewer: Evaluate the user specified in workflow argument -reviewer of TC OOTB action handler EPM-fill-in-reviewers used in preceding Review Task.
  ResponsibleParty (default): Evaluate the user specified in workflow argument -assignee of TC OOTB action handler EPM-auto-assign used in preceding Do Task.

  -TimeWindowEnd
  Sets the job attribute Time Window End (Format [hh:mm:ss]) to define the time of last possible execution of the job per day. If not set default value is 00:00:00 meaning the job may be executed at any time.

-TimeWindowStart
Sets the job attribute Time Window Begin (Format [hh:mm:ss]) to define the time of first possible execution of the job per day. If not set default value is 00:00:00 meaning the job may be executed at any time.

- - -

- **T4X-manage-Connection4Session**
  This handler manages preferred connection settings while a workflow is executed. It stores the current settings when called with "-Mode=SAVE" and restores these settings when called with "-Mode=RESTORE". Option "-ProductKey=..." can specify a certain T4x product or type of connections for which the preferred connection is supposed to be (re-)stored. If it is not specified, preferred connections will be (re-)stored for all T4x products within the installation.
  The handler can be used in any task, preferably in actions "Start" and "Complete".

  > Note:
  >
  > A change in preferred connection settings outside the workflow while this is still not completed may be changed again by the workflow's preferred connection RESET on workflow completion.
  >
  > The EA logon data must be present and correct. This is best checked with the "T4X-validate-EALogon" rule handler before the T4X-manage-Connection4Session is called with the -Mode=SAVE argument.

  -Mode
  Possible values are:
  SAVE - saves the current EA login into a product specific proxy.
  RESTORE - restores the EA login from a product specific proxy.

  -ProductKey
  (optional) The parameter can specify a T4x product, e.g. T4S. If a product is specified, the action is only performed for the specified product, resp. type of connection.

  - - -

- **T4X-transfer-GenericObject4TargetType**
  No mandatory arguments, all handler arguments are optional.

  -add_related_objects
  If set to TRUE, the handler adds related object information as simulated BOM lines to the TcData buffer during the BOMView based data extraction. This means it combines the BOMView based with the BOM4REL based data extraction.

  -bom4rel_use_bomview
  Requires -useBom4Realation4Transfer=TRUE. If parameter is set to TRUE, a BOM transfer based on BOM view information is done if BOM view exists.

-Bom4RelationTargetTypeName
Required for BOM transfers based on relation information. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-BomHeaderTypeList
Required for BOM transfers. The parameter specifies the variable part of the preference names that is used to identify which structure needs to be transferred.

-ChangeType
The parameter defines the preference prefix ([ProductKey][ChangeType]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item,...).

-ChangeTypeName
The parameter defines the preference prefix ([ProductKey][ChangeTypeName]TypeList) to identify if the target should be search with a Teamcenter Change object and its folders (e.g. Solution Item, Problem Item, ...).

-check_EPM_Targets
If set to TRUE, the handler reads data from each Workflow job target data additionally.

-ProductKey
The parameter specifies the T4x product, e.g. T4S.

-use_EA_system
If set, T4X will use the EA system stated as the value of this argument.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

-use_EPM_objects_as_targets
With this parameter it is possible to decide if the root_task (job) or the current_task of a workflow in the inbox is used as transfer object. Possible values are: root_task and current_task. Instead it is also possible to use the EPM_target_attachment (default) In addition the preference [ProductKey]_[TargetTypeName]TypeList needs to contain the type EPMTask.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

-useBom4Relation4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on relation information.

-useView4Transfer
Required for BOM transfers. If set to TRUE, a BOM transfer is enabled based on BOM view information and the TCL namespace for the mapping switches from GENOBJ to GENBOM.

- - -

-bom_rev_rule
This parameter specifies the TC BOM revision rule name. If set, the stated revision rule overrides the BOM revision rule setting [ProductKey]_BillOfMaterialMapping4 in the Teamcenter preferences. If you use a BOM rule handler as well, it needs the same argument.

-bomview_prio_list
This parameter specifies [list of BOM view types]. Value is a comma-separated list of BOM view types; if set, T4x reads this list and checks if one of the stated BOM view types is present in the target according to the preference [ProductKey]_BillOfMaterialTypeList. Then it uses the Teamcenter BOM object according to the first matching type only and ignores all the others given to this handler. If none of the stated types is present, no BOM is transferred although the target list may contain several BOM views. The value for this argument is a comma separated list without additional blanks. T4x just removes the commas and uses the strings as they are as type names, e.g. "-bomview_prio_list=view,MBOM,BOM3"

-hide_GDELines
If set to TRUE, T4x will hide GDE BOM positions.

-hide_incremental_changes_bomlines
If set to TRUE, T4x will hide incremental changes.

-hide_substitutes
If set to TRUE, T4x will hide substitutes.

-hide_suppressed_bomlines
If set to TRUE, T4x will hide suppressed BOM positions.

-hide_unconfigured_bomlines
If set to TRUE, T4x will hide unconfigured variants.

-hide_variants_bomlines
If set to TRUE, T4x will hide variants.

-no_transfer_of_empty_bom
If set to TRUE, an empty Teamcenter BOM (it does not contain any position) will be ignored; else T4x will try and create an empty BOM in the foreign system, i.e. the BOM header with all its data but no BOM position. Caution: this is only used for BOMs that are stored empty in Teamcenter; if

a BOM gets empty during the mapping because all its positions are skipped, this setting will not skip the BOM transfer; that has to be handled in the BOM mapping separately

-scan_max_bom_level
This parameter defines the maximum number of structure levels that are read by T4x. The default value is 1.
Only relevant if multi-level BOM transfer is configured.

-skip_bomlines_by_condition
If set to TRUE, T4x will skip BOM positions based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]SkipCondition4[BOMLineType]

-skip_bomlines_by_condition_prefix
This parameter overwrites the default preference name used by parameter -skip_bomlines_by_condition.

-skip_unconfigured_bomlines
If set to TRUE, T4x will skip unconfigured BOM positions.

-unpack_all_bomlines
If set to TRUE, T4x will provide all BOM positions unpacked and allow the packing in the mapping according to any logic defined there

-unpack_bomlines_by_condition
If set to TRUE, T4x will provide all BOM positions unpacked based on a defined set of BMIDE conditions.
The validated conditions are defined in a site preference with the following naming convention: [ProductKey]_[TargetTypeName]UnpackCondition4[BOMLineType]

-use_attached_rev_rule
If set to TRUE, this BOM handler will use the Teamcenter BOM revision rule which was attached to the workflow target by the additional T4x handler T4X-attach-RevisionRule. Additionally this functionality needs the following preference set: [ProductKey]_BillOfMaterialType_RevisionRuleAttachmentPath=[Relation]:RevisionRule The name part BillOfMaterial depends on the Teamcenter BOM type for which that configuration should be valid, so instead of the default name BillOfMaterial that name part may be for example OrderBillOfMaterial or EquiBillOfMaterial The [Relation] stands for the path that is used to find the RevisionRule, e.g. IMAN_reference or if the revision rule is attached to the workflow itself then EPM_target_attachment

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]

For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision
or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error
If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

- **T4X-transfer-ProductConfigurator**
  This handler transfers objects of Teamcenter Product Configurator to an external system.

  -ProductKey
  The parameter specifies the T4x product, e.g. T4S.

  -revision_rule
  This parameter specifies the revision rule name that should be applied to the product configuratior context during the transfer.

  -revision_rule_preference

This parameter specifies the name of a preference that configures the revision rule name that should be applied to the product configuratior context during the transfer. This approach allows a revision rule name change without changing the workflow template.

-revision_rule_relation
This parameter specifies the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer.

-revision_rule_relation_preference
This parameter specifies the name of a preference that configures the relation from the transfer target object to an attached revision rule that should be applied to the product configuratior context during the transfer. This approach allows to change the relation to the revisions rule without changing the workflow template.

-use_EA_system
If set, T4X will use the EA system stated as the value of this argument.

-use_EBS_system
If set, T4O will use the Oracle EBS system stated as the value of this argument.

-use_SAP_client
If set, T4S will use the SAP client stated as the value of this argument.

-use_SAP_system
If set, T4S will use the SAP system stated as the value of this argument.
If SAP client and SAP system are defined using Teamcenter preferences the syntax is -[argument]=usePreference[:group]:[PreferenceName].
E.g. specify -use_SAP_system=usePreference:T4S_SAP_System in case of site preference or -use_SAP_system=usePreference:group:T4S_SAP_System in case of group preference.
As soon as a T4S workflow action handler with the argument "-use_SAP_system" and "-use_SAP_client" was executed, T4S will go on using this SAP connection until another connection is defined.

- - -

-AddObject4Mapping
This parameter allows T4x reading additional Teamcenter data during the transaction.
Syntax: -AddObject4Mapping=[EPM_attachment_type]:[ObjectType][:[RelationType]:[ObjectType]][:PropertyName]
For [EPM_attachment_type] use value EPM_target_attachment, EPM_reference_attachment, EPM_signoff_attachment or EPM_release_status_attachment.
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
to read attribute values of a TC object in the mapping that have not been configured (i.e. set in the corresponding Mapping4 preference).
Specify e.g. -AddObject4Mapping=EPM_target_attachment:ChangeNoticeRevision

or -AddObject4Mapping=EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
to read attribute values of a TC object e.g. added via action handler EPM_attach_related_objects to workflow task or manually in the "New Process Dialog".
In case of several needed objects the syntax is -AddObject4Mapping=usePreference:[PreferenceName].
The preference [PreferenceName] contains the definition for all needed objects,
e.g. -AddObject4Mapping=usePreference:T4X_AddObject4MappingPref with
T4X_AddObject4MappingPref =
EPM_target_attachment:ItemRevision:CMHasSolutionItem:ChangeNoticeRevision
EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info:#__getAllProperties__#:Properties
In mapping use function ::T4X::TC::MAPPING::RootTaskFieldMapping to read the object attribute value,
e.g. set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ItemRevision:CMHasImpactedItem:ChangeNoticeRevision" "object_desc"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_target_attachment:ChangeNoticeRevision" "object_name"]
or set AttrValue [::T4X::TC::MAPPING::RootTaskFieldMapping "RootTask:EPM_reference_attachment:ItemRevision:IMAN_reference:T4S Status Info" "T4S_Free1"]
Specify -AddObject4Mapping=root_task:EPMTask to read workflow task attributes in the mapping.
In mapping get the task attribute from the TcData buffer, e.g. set GovClassification [::T4X::TC::MAPPING::RootTaskFieldMapping RootTask:root_task:EPMTask gov_classification]
(Check preferences T4X_Property2IgnoreList4EPMTask and T4X_Property2ProcessList4EPMTask for the attribute and adapt them accordingly).

-TargetTypeName
The parameter specifies the preference prefix for the preference names (e.g. [ProductKey]_[TargetTypeName]TypeList) that is used to identify which structure needs to be transferred.

- - -

-useSpecialMode
If set to REVERSEMAPPINGONLY, the handler skips the transfer after the mapping and calls directly the reverse mapping procedure.

- - -

-collect_all_errors
If set to TRUE, all attached workflow targets are processed without stop until all targets are processed. Then the whole set of errors can be handled together.

-continue_on_error

If set to TRUE, the handler does not stop in case of an T4x error (i.e. Teamcenter error number range 212000 to 212999 which is reserved for T4X).

-on_error_set_task_result
If set to TRUE, the parameter enables the update of the EPM Task result property in the error case.

-on_error_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was not successful. If not specified the value "Unable_to_complete" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-on_ok_use_task_result
This parameter defines the string that is stored to the EPM Task result property if the transfer was successful. If not specified the value "Completed" is used. Prerequisite is that the argument -on_error_set_task_result=true.

-target_condition_preference
With this parameter T4x will use the conditions defined by the specified preference to validate the BMIDE condition against the selected target object.

-target_condition_preference_prefix
This prefix is extended by the object_type name to generate the final preference name to retrieve the condition name [Preference Prefix][TransferObjectTypeName].

- - -

B. Glossary

A

ABAP
ABAP is a proprietary programming language of the SAP AG.

Admin
is the term used in this document for people who install and configure Teamcenter and its components. This is in contrast to the "user" role.

Admin UI
Web based administrative user interface of the GS and BGS.

AIG
The entire Active Integration Gateway product family.

AIG_ROOT
Please see **GS_ROOT** and **BGS_ROOT**. This term is used if something is true for both the GS and BGS.

AI-Object
Application-Interface Object

API
Application Programming Interface.

Apps
See "GS".

AppServer
Application Server.

B

BAPI
The Business Application Programming Interface allows external programs to access objects and business processes in SAP.

BGS
Basic Gateway Service.

BGS_ROOT
The installation directory of the Basic Gateway Service (e.g. *C:\Siemens\BGS*).

BMIDE
Teamcenter Business Modeler IDE (Integrated Development Environment)

BOM
A Bill Of Materials is a list of the parts or components and their quantities that are required to build a product.

BOM Header
A BOM Header is the top item of a BOM. BOMs can have multiple levels, so this often means the top item of the actual level.

BOP
The Bill Of Process describes a manufacturing process and lists the operations and steps with all their instructions, consumed materials, resources, work places and machines.

C

CCObject
Collaboration Context Object

CEP
Camstar Enterprise Platform

Change Master
The Engineering Change Master (ECM) contains the metadata to a change number.

Characteristic
An characteristic is an attribute of a SAP class.

CIO
Camstar Interoperability

CLM4S
Closed Loop Manufacturing for SAP S/4HANA®

D

Data Carrier
Please see **Vault**.

Dataview
The Dataview is an extension to the Teamcenter RAC and is deployed as part of the TEM installation process of the Teamcenter Gateway. The Dataview is used to display the real-time data of external applications, associated with Teamcenter objects.

Dataview mark-up
is the language understood by the Dataview. The Dataview receives messages written in this language from the T4x server. Such messages can be formatted as XML or JSON. Normally users do not see such messages. They may however appear in log files or error messages. The so called prop mapping (e.g. *t4s_prop_mapping_template.sd*) contains TCL commands that compose messages in the Dataview mark-up.

DCD
Data Collection Definition

DIR
DIR is the abbreviation for a SAP Document Info Record.

Document Key
A Document Info Record is identified by the combination of Document Type, Document Number, Document Part and Document Version.

Document Structure
A Document Structure is like a Bill Of Materials for Documents.

E

EA
stands for Enterprise Application, any software or set of computer programs used by business users to perform various business functions in context of current integration's portfolio with Teamcenter.

ECN
The Engineering Change Notice can also be called an Engineering Change Note, Engineering Change Order (ECO), or just an Engineering Change (EC).

EPM
Enterprise Process Modeling

EWI
Electronic Work Instructions

F

File Stream
Method of transfer to send an original to SAP.

G

Gateway Menu
An additional menu item of the Teamcenter Gateway software available in the Teamcenter RAC.

GRM
The Generic Relationship Management provides a general way in which two objects can be associated via a relationship.

GS
Gateway Service, manages the communication between Enterprise Applications.

GS_ROOT
The installation directory of the Gateway Service (e.g. *C:\Siemens\GS*).

GUI
Graphical user interface.

GUID
Globally Unique Identifier

I

IDGEN
The IDGEN is a mechanism to get an external ID from the ERP system when assigning a Teamcenter ID.

Inspection Plan
Contains characteristics to be inspected in an operation and equipment to be used.

iPPE
Integrated Product and Process Engineering is a module that can be used to mange products with many variants.

ITK
The Integration Toolkit (ITK) is a set of software tools provided by Siemens PLM Software that you can use to integrate third-party or user-developed applications with Teamcenter.

J

JCO
The Java Connector is an interface to . In the context of it is now mostly replaced by the Netweaver RFC interface.

JDBC
Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database.

Job
Teamcenter Gateway features asynchronous transfer. This datatransfer is managed via a Job.

Job Pool
The Job Pool contains all finished and unprocessed Jobs. It is managed by the BGS.

Job Server
The Job Server on the Basic Gateway Service (BGS) manages the Job and distribution them to the Job Agent for processing.

JSON
JavaScript Object Notation is a lightweight data-interchange format[1].

K

KPro
Kpro stands for Knowledge Provider. See also Data Carrier.

L

LOV
List of Values

M

Mapping
The mapping is part of the T4x configuration. It contains the code that controls the behavior of the data transfer between Teamcenter and the ERP system.

1 JSON.org

MFK
Multi-key functionality in Teamcenter.

MM
MM is the abbreviation for a SAP Material Master.

MOM
Manufacturing Operations Management

N

NCN
Non-Conformance Notification

NetWeaver RFC SDK
The NetWeaver RFC SDK contains libraries for 3rd party applications to connect to . It can be obtained from the SAP ONE Support Launchpad.

O

Object Key
The Object Key is a string that contains the ID of an Enterprise Application object. If the identifier is a combination of multiple keys, then the Object Key is a combination of those keys in a defined order and format.

Object Link
A relation between SAP objects like Material Master and Document Info Record.

Object Management Record
Belongs to a SAP Change Number and Documents changes of one particular SAP object like a Material Master.

OOTB
Out of the box

Original
A representation of a file in SAP.

OSS Note
The OSS Note is an online patch service for SAP. The patch can be identified by the OSS Notes number.

P

PIR
PIR is an abbreviation for a SAP Purchase Info Record.

Portal Transaction
This means that a transfer to SAP that is not triggered by a workflow handler but via the Gateway Menu.

R

RAC
stands for Rich Application Client also referred to as rich client or portal.

Revision Level
Used to show changes with reference to a change to a SAP Material Master or Document Info Record.

RFC
Remote Function Call (SAP)

S

SAP
SAP S/4HANA® / SAP Business Suite®

SAP GUI
This is the application for the SAP Business Suite® and SAP S/4HANA®.

SAP Logon
This is the application that a user needs to start the SAP GUI for a particular system. It may also refer to the process of logging in to SAP in Teamcenter via .

SAP Portal iView URL
Can be used to show sap content in a browser window.

Session Log
Shows one log file for each Teamcenter session. Written if T4x transactions are executed

SSL
Secure Sockets Layer.

T

T4O_ROOT
Please see GS_ROOT

T4S 4-Tier Client (SAP Lite)
The 4-Tier Client or SAP Lite is a stripped down GS. It´s only purpose is to open the SAP GUI on a Teamcenter 4-Tier Client.

T4x
The entire Teamcenter Gateway product family.

TAO
The ACE ORB is a open-source and standards-compliant real-time C++ implementation of CORBA based upon the Adaptive Communication Environment (ACE).

TargetTypeName
This is the T4x internal name for the transaction type. E.g. `MaterialMaster` or `DocumentInfoRecord`.

TC
Teamcenter

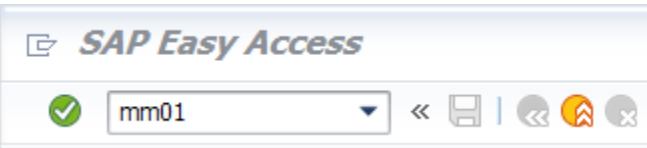
TCL
is a high-level, general-purpose, interpreted, dynamic programming language.

TCPCM
Teamcenter Product Cost Management

TCPCM4S
Teamcenter Product Cost Management Gateway for SAP S/4HANA

TEM
Teamcenter Environment Manager

Transaction Code
A Transaction Code is a quick access code for a Transaction in the SAP GUI:



Transaction Log
The Transaction Log is a T4x logfile on the BGS. It contains log information for a specific T4x transaction.

Transfer Window
The Transfer Window triggers transactions via the Gateway Menu.

Transport Package
A file that contains functions that can be imported to SAP.

U

UOM
UOM stands for Unit of Measure.

URI
Unified Resource Identifier: a generalized from of a resource locator (URL) and resource name (URN), which just identifies a resource, but is not necessarily sufficient to locate (find) the resource. URIs are often used to identify configurations in Java and other languages. See **https://en.wikipedia.org/wiki/Uniform_Resource_Identifier** for more details.

URL
Unified Resource Locator: a string with a certain format, allowing to load a resource from a network. URLs are a specific form or URNs.

User Exit (SAP)
A User Exit is a code for a program that is called if an object like an MaterialMaster has been changed or updated. In the context of T4S it is often used to initiate the process to trigger a transfer from SAP to Teamcenter.

User Log
The User Log is a T4x logfile on the BGS. If you define a customized logchannel, the information is written into a User Log of that name.

V

Value Set
A Value Set is the SAP term for a list of selectable values for a characteristic.

Vault
The Vault is a server where a SAP DocumentInfoRecord original is stored. A synonym is also Data Carrier.

W

WBS
WBS is an abbreviation for a SAP Work Breakdown Structure.

X

XML
Extensible Markup Language is designed to store and transport data in a format that is both human- and machine-readable.

XRT
stands for XML Rendering Template, also known as XML Rendering Stylesheet. These are XML documents stored in datasets that define how parts of the Teamcenter user interface are rendered. They are used for the Rich Client as well as the Active Workspace.

Z

ZPTC
This is the short name for a Z-Table with the name /TESISPLM/ZPTC, used to trigger a transfer from SAP.

Z-Table
"Z" is a well-known prefix name for custom tables in the SAP world. A special table used with is the table `/TESISPLM/ZPTC`.

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.