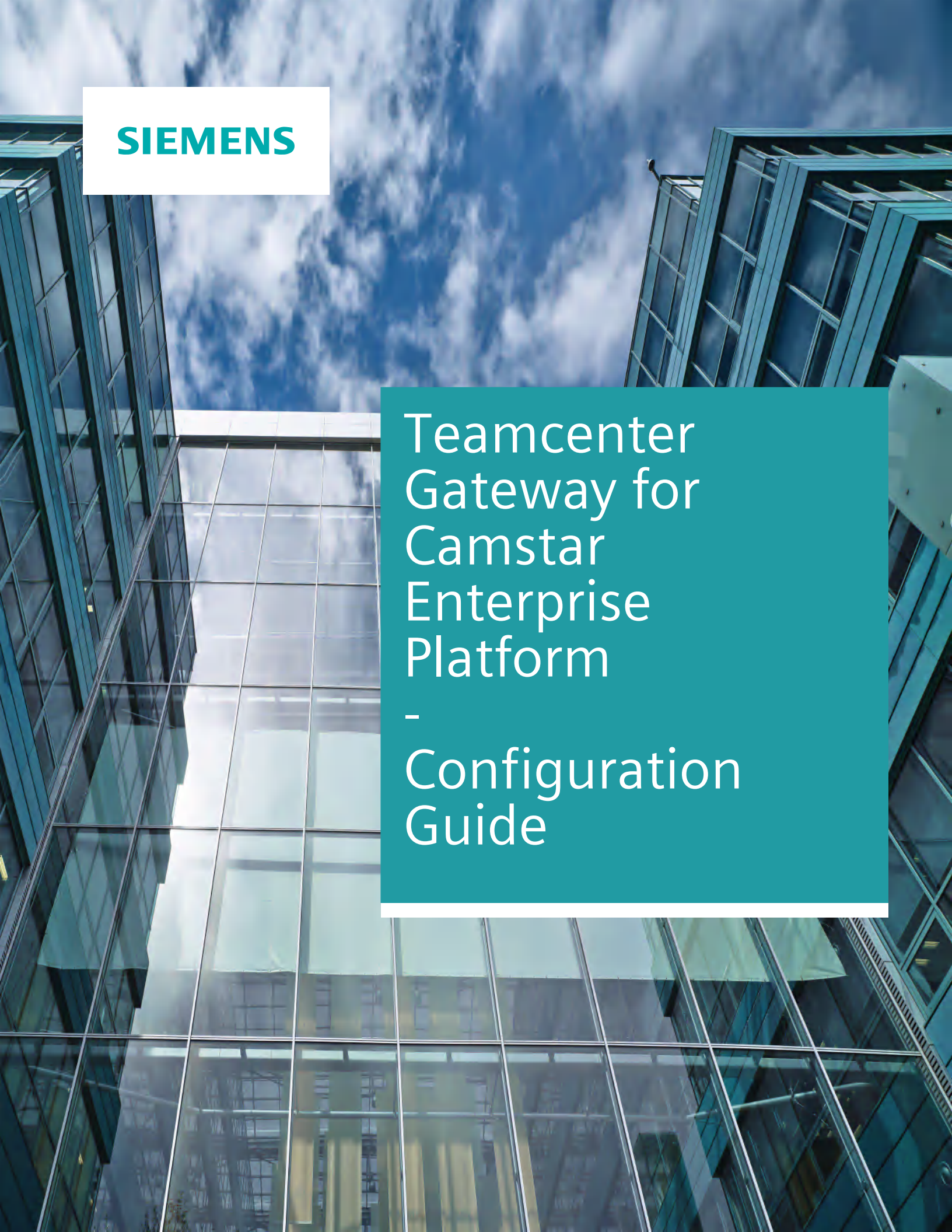


SIEMENS

Teamcenter Gateway for Camstar Enterprise Platform - Configuration Guide

Contents

Manufacturing BOP Transfer

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: December 2019

1. Introduction

The Teamcenter Gateway for Camstar Enterprise Platform **(T4CEP)**software solution is an integration software that provides data and process integration between Teamcenter® by Siemens Product Lifecycle Management Software Inc. and Camstar Enterprise Platform by Siemens Product Lifecycle Management Software Inc..

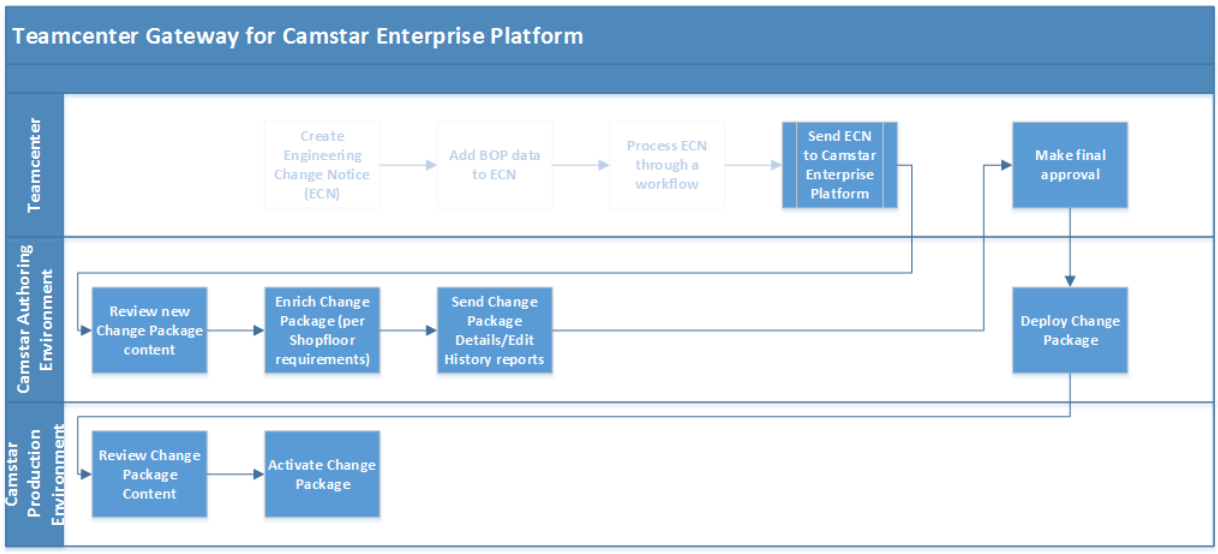
T4CEP provides a wide range of automatic and workflow functions to transfer and synchronize data between Teamcenter and Camstar Enterprise Platform.

This document details the components of Teamcenter - Camstar integration, which are available out of the box and configurable to meet customer specific solution requirements.

2. T4CEP and CIO/MIO Interaction

Integration Overview

T4CEP relies on the Change Management module to transfer data with CEP. Following is a depiction of an integrated change management process, where Change Notice from Teamcenter is downloaded along with its contents to the Change Package in CEP for review or deployment to the Production Shop Floor.



Following sections of the document will detail the technical aspect and functional use cases supported by the integration.

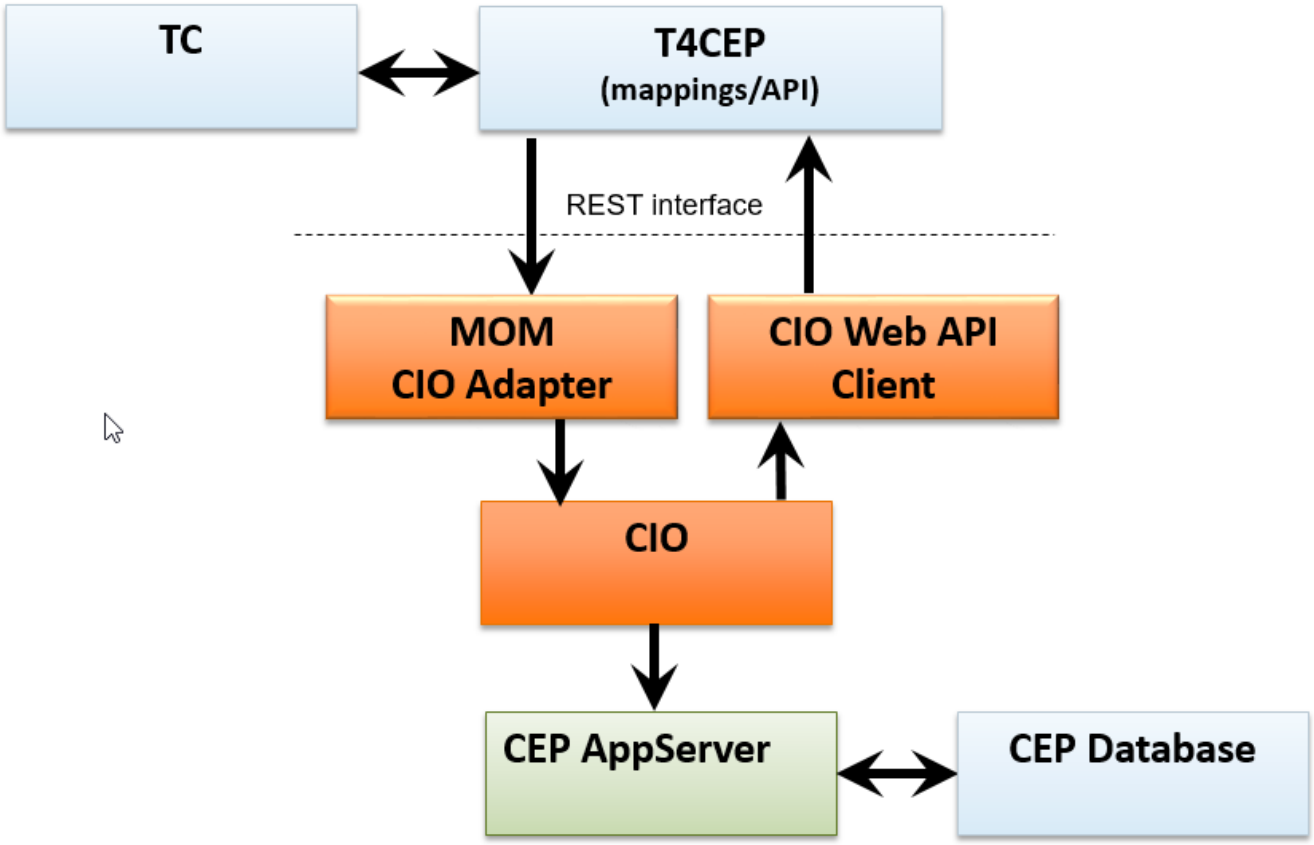
Basic Components

Teamcenter to Camstar Enterprise Platform (CEP)

- Invoke the T4CEP transaction by a Teamcenter workflow.
- T4CEP gets data from Teamcenter and transform to XML.
- T4CEP uses the MOM CIO (Camstar Interoperability) MIO (Manufactoring Interoperability) Adapter, which abstracts the CEP APIs into a generic web interface.
- CIO/MIO is configured with adapters and maps, and communicates with the CEP AppServer, which in turns reads/writes the CEP Database.

CEP to Teamcenter

- CIO/MIO sends non conformance message or request for approval to T4CEP.
- T4CEP transforms XML to dictionary and create object in Teamcenter or starts workflow



3. Basic Configuration

The File t4cep_mapping_config.sd

The file *t4cep_mapping_config.sd* in the directory *<GS_ROOT>\var\mmap\t4cep_mapping_config* is the first file that is read by each TCL worker thread or process of the T4CEP software. It loads the rest of the mapping files and may contain basic settings as well as variables which are supposed to be used in more than one mapping file.

The following basic configuration data can be set in generic mapping configuration file *t4cep_mapping_config.sd*:

- System login data, see chapter **Provide CEP Account Information** in the **Active Integration - Installation Guide**

- `::ITK::setCredentialsAlias` allows to set the alias to be used to make the Teamcenter connection ("Default@Teamcenter" by default, but this can be changed). The credentials for this alias must be set using the script `itk_connect2db.tcl`.

- `::T4X::CONNECTION2EA::selectActiveConnection2EA`, `::T4X::CONNECTION2EA::readConnectionInfo4Session` and `::T4X::CONNECTION2EA::setCredentialsAlias4UseCase` are used to setup the connection(s) to the connectivity layer of the Camstar Enterprise Platform. You can specify the connection URL and by addressing a specific endpoint you implicitly also specify the authentication method. You can also provide credentials for a technical user which T4CEP uses to authenticate and set a default connection to use, if no other connection has been specified, e.g. in the workflow. To manage credentials use the *connect_to_cio.tcl* script.

- The variable `::TargetTCClientPoint` determines the URL T4CEP will transfer to CEP to refer back to T4CEP.

- The variable `::TargetTCFileShare` determines where T4CEP stores files which should be transferred to CEP. Usually this denotes a Windows share accessible from both sides. The variable `::SourceTCFileShare` is the static portion of the link for the same files in a format CIO accepts, so CEP can access the files on their side. The variable `::CamstarTransferArea` is used for intermediate file storage and is automatically cleaned up.

  > Caution:
  >
  > Provide credentials before starting document transfer. Contact your system administrator to make sure that the credentials known in the user session, e.g. if the T4CEP is running via Windows Services.

- The variable `::CamstarUI` is the URL by which Teamcenter can open a view on Camstar Enterprise Platform, e.g. for a change package.

- Sourcing of mapping files with `source -relax`.
  Example: `source -relax t4cep_object_mapping.sd`. This means T4CEP has to read the content of the file *t4cep_object_mapping.sd*. The argument `-relax` means the file is looked up in the memory first and only if its content is not there, the file is loaded from disc (only in the same directory *<GS_ROOT>\var\mmap\t4cep_mapping_config*).
  In principle, the file names may be modified freely as long as the file extension *.sd* is kept. Every file stated there that has the file extension *.sd* is actually used for the mapping functionality. In fact the only file T4CEP actually uses for the mapping is the compiled mapping file *t4cep_mapping_config.rfdt*. So this is what you need to think about in order to create the compiled mapping file correctly.
  In order to not use a mapping file it is enough to not "source" it in *t4cep_mapping_config.sd*. However, we strongly recommend keeping only those mapping files in the *mmap* directory that you really want to use!
  Be sure to have the correct file names (the files located in *<GS_ROOT>\var\mmap \t4cep_mapping_config*) in the "source" section of this file

Mapping Templates

T4CEP provides mappings for basic functionality in the folder *<GS_ROOT>\var\mmap \t4cep_mapping_config*. For these mappings to work, an OOTB data model is assumed on both sides (Teamcenter and Camstar Enterprise Platform) with some necessary extensions on Teamcenter side. You must install the template in *<GS_ROOT>\var\template\cep0gateway* for T4CEP to work correctly. The cep0gateway must also be installed in production environments. An example for additional extensions is contained in *<GS_ROOT>\var\template\t4cepdemo*. This latter template is not meant to be used in production, it contains some shortcuts or simple solutions and is just for demonstration and to guide customers when building their own custom templates.

The samples in the mapping template files may serve as a guideline to your own customer specific implementation and show the use of the functions provided to make the implementation easier.

- *t4cep_bom_mapping.sd*
  Sample implementation of an obsolete, simple TcData-based (non-PLMXML-based) export of a TC BOM to CEP. The TC BOM occurrence note `T4CEOperation` is evaluated to determine the CEP operation name. This transfer is not be used in conjunction with the real, PLMXML-based routing transfer!

- *t4cep_connection_mapping.sd*
  Contains basic T4CEP connection procedures and settings necessary to make the T4CEP connection mapping work.

- *t4cep_custom_data_mapping.sd*
  Contains basic adaptions to the custom data model, e.g. used TC dataset types and the relations used, CM folder mapping. This file will grow with future releases to contain more custom specific adaptions.

- *t4cep_custom_import_services.sd*
  Implementation of inbound service calls from CIO. Usually no custom changes should be necessary in this file.

- *t4cep_custom_services.sd*
  Implementation of outbound service calls from TC to CIO. Usually no custom changes should be necessary in this file.

- *t4cep_object_mapping.sd*
  Implementation of the export of non-structured objects (Materials, DocumentSet, Document, Change Package) from TC to CEP.

- *t4cep_plmxml_object_mapping.sd*
  Implementation of the PLMXML-based routing export from TC to CEP. This includes the transfer of TaskLists, ERPBOM, ERPRoute, DCDs, Workflow, BillOfProcess and Resources in 150% and 100% configurations.

- *t4cep_typehandling.sd*
  Implementation of a centralized mapping of CEP attribute names to CEP types.

4. Change Transfer – Change Package

Create Change Package

Capability to create Change Package in Camstar Enterprise Platform, based on the ECN released from Teamcenter.

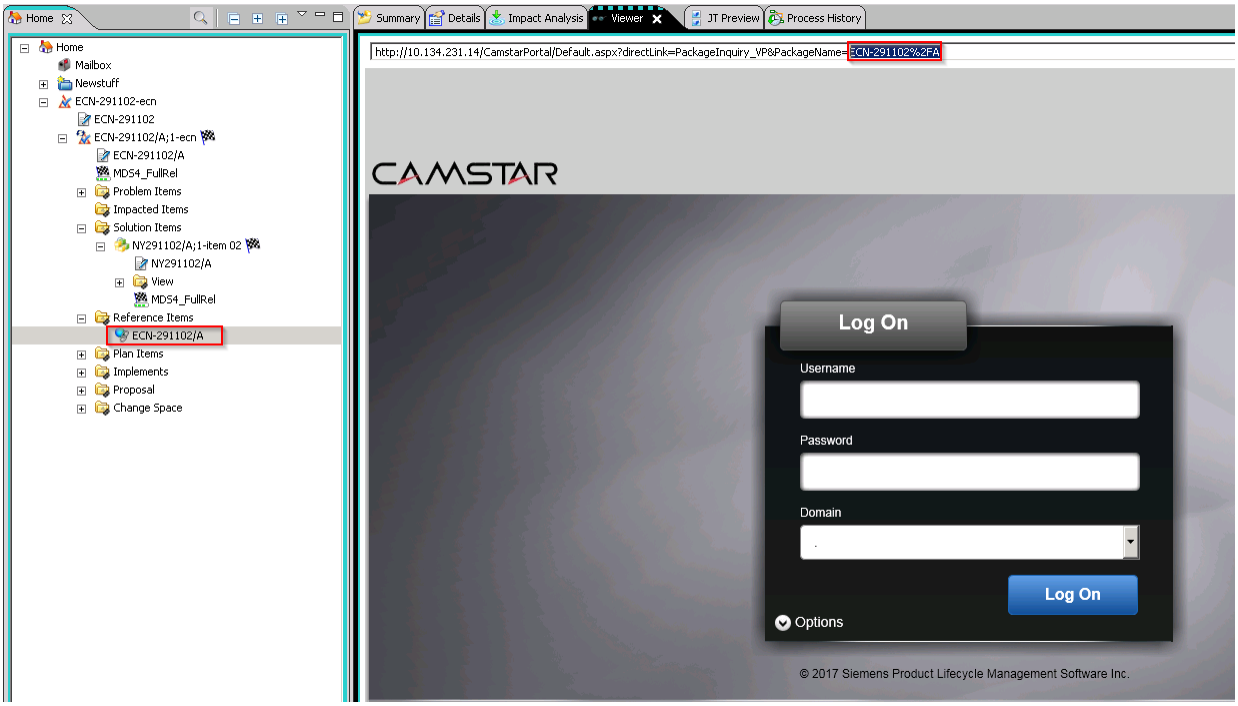
When creating Change Package for the first time in CEP, the following sequence of transactions are executed:

1. CEP Change Package Start – Transaction to be executed only once in the lifetime of a CEP Change Package. If the Change Package already exists, the `StartChangePkg` transaction end with status `UPDATE`. In the next step T4CEP executes query transaction to get the `momWorkflowStep.Name` from CEP. The Change Package can be enriched in the step `Download In Process`.

2. CEP Change Package contents setup – One or more transactions executed to transfer the Solution Items content found in the ECN being released from Teamcenter.
   In case of any failures encountered during the transfer of ECN contents, the Change Package will remain at the "Download in Progress" step in CEP.
   In such a scenario, the ECN can be re-released from Teamcenter without the need to execute CEP Change Package Start transaction again.

3. CEP Change Package Move Standard – Transaction executed once all the Change Package contents have been setup in CEP.

To create a ChangePackage in CEP, transfer a TC ChangeNoticeRevision with the workflow handler T4X_transfer-Generic-Object4TargetType with the argument `-position=StartChangePkg` and `-TargetTypeName=ChangeNotice`. The procedures in the TCL namespace `::T4CEP::GENOBJ::CUSTOM::MAPPING` use the TargetTypeName `ChangeNotice` and the argument value `StartChangePkg` to branch into the corresponding implementation of the mapping. The procedure `createObject` is used to create a TC URL object which references the ChangePackage in the web user interface of CEP.



Query Change Package Status

Capability to request Status of a Change Package from CEP. This can be leveraged to understand whether or not a Change Package exists in CEP and, if found, receive its Status (0 for Closed, 1 for Open, 3 for Rejected, 4 for Voided, 2 for Deployed and 5 for Approved).

The relevant field is called `momPackageStatus` and this is also the name of the T4CEP workflow handler argument. The procedure `::T4CEP::GENOBJ::CUSTOM::MAPPING::performChangeStatus` can be used to execute the CIO call.

Request for Change Package Approval from Teamcenter

Capability to notify Teamcenter about Change Package updates made in CEP, if any, and allow Manufacturing Engineering to perform sign off in Teamcenter so that they can be released to the Shop Floor for Production.

It is implemented by the T4CEP web service MOMECNImport (usually accessible via a URL similar to https://<GS-hostname>:11301/pxml/momecnimport and implemented in the TCL procedure `::T4CEP::CUSTOM::IMPORT::SERVICES::MOMimportService`). The requested status from CEP is an integer and it is mapped to workflow name in the TCL dictionary `::T4CEP::CONFIGURATION::StateWorkflowMap`. T4CEP will start the corresponding TC workflow based on the incoming change specification and status information. This allows an easy adaption to custom requirements.

Approve Change Package

Capability to move the Change Package to Production step in CEP, based on review process signoff in Teamcenter.

This capability is implemented by setting the change package status in CEP to "Approved". In the demo workflows this happens in the "T4CEP_Review" example workflow if the TC Reviewer approves his task.

Reject Change Package

Capability to move the Change Package to Reject step in CEP, based on review process signoff in Teamcenter.

This capability is implemented by setting the change package status in CEP to "Rejected". In the demo workflows this happens in the "T4CEP_Review" example workflow if the TC Reviewer does not approve his task and decides that improvements have to be made in CEP.

5. Document Transfer – Document

Create Document

Capability to create Document in CEP, with a URL reference pointing to a file stored on the file server with the DataSet extracted from Teamcenter. In case of MEOperation and MEProcess Revisions, the Textual Work Instructions (TWI) in PDF format and / or 3D JT file as well as the 3D Snapshot representation in JPG format are transferred as Documents to CEP.

Please note that current configuration includes only following types:

```
variable FileHandlingDatasetMap [dict create]
    dict set FileHandlingDatasetMap PDF              PDF_Reference
    dict set FileHandlingDatasetMap JPEG             JPEG_Reference
    dict set FileHandlingDatasetMap SnapShotViewData Image
    dict set FileHandlingDatasetMap Mes0MEWIPreview  Mes0PDF
    dict set FileHandlingDatasetMap DirectModel      JTPART
```

To support other file formats for example for GIF format on the ItemRevision, please do the following steps:

- Add desired file format as value to preference as followed:

```
T4CEP_DocumentMapping4ItemRevision =
IMAN_specification:GIF:ref_list/GIF_Reference:ImanFile
IMAN_specification:GIF:#__getAllProperties__#:Properties.
```

- Add the entries into Tcl dictionary FileHandlingDatasetMap in *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_custom_data_mapping.sd* to support the new file format:

```
dict set FileHandlingDatasetMap GIF GIF_Reference.
```

> Caution:
>
> If you choose the shared folder as a target for storing the files, the shared folder has to exist and to properly mapped (write access rights required) before you start the workflow in Teamcenter.

Please note that the relation IMAN_specification to the DataSet is set in the variable `ReferenceTypeHandling`. Depending on configuration T4CEP stores the file from Teamcenter in the shared folder or you can use the file server as well. By default, T4CEP supported the DataSet for the Document Revision as well. The Document Revision can be linked to the Item Revision. This structure allows to use the same Document Revision for different Item Revisions. It can be for example legal regulations or standards, which are the same for all Product Revisions. In this case T4CEP can retrieve the structure on the lined object and transfers the files from the DataSet refereed to both objects.

Additionally, during Manufacturing BOP transfer, the capability is leveraged to create / update the EWI URL against the CEP Task as a Document. If any DataSets exists on the MEProcess or MEOperation (Mes0MEWIPreviewRelation), T4CEP transfers a Document. In this case, the attribute `EWIURL` is empty. During the BOP Override transfer, T4CEP starts the Document transaction again and populate the XML attribute with the value from `ewi0step_url`. This URL points to the Teamcenter MEOperation or MEProcess in Active Workspace.

The dictionary key `Transfer` manages the transferring:

```
dict set CamstarInputDat Transfer TRUE
```

If the `Transfer` does not equal `TRUE`, T4CEP do not transfer any documents.

Maintain Document fields

The mainained fields are the keys of the `CamstarInputDat`. Go to the `::T4CEP::GENOBJ::CUSTOM::MAPPING::TC_Object2EA_Object` procedure in *GS_ROOT//var/mmap/t4cep_mapping_config/t4cep_object_mapping.sd* to get more details.

6. Document Set Transfer – Document Set

Create Document Set

Capability to create Document Set, so that the collection of one or more Documents created against a CEP Product or Task can be cataloged for reference.

With `if` statement in the T4CEP mapping file for objects:`if {$TargetTypeName eq "DocumentSet"}` starts the building up of the `CamstarInputDat` dictionary. The DataSet relation as well as the file type are configured in the following variables:

```
[dict get $::T4CEP::CONFIGURATION::ReferenceTypeHandling]

[dict get $::T4CEP::CONFIGURATION::FileHandlingDatasetMap]
```

Please refer to the *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_custom_data_mapping.sd* to configure additional file type or relation. Do not forget to extend the preferences according to expected file type or relation.

The transaction starts in `performTransfer` by calling:

```
if {$TargetTypeName eq "DocumentSet"} {

        if {[dict get $CamstarInputDat Transfer] eq "TRUE"} {
          dict set CamstarInputDat Action Sync
          set Status
[::T4CEP::CUSTOM::SERVICES::camstarDocumentSetService \
            $CamstarInputDat]
        } else {
          ::T4X::TRANSLOG::writeCustomMappingLog $TransactionId SKIPPED \
            "Camstar service call for DocumentSet transfer skipped"
          set Status "SKIPPED"
        }

      }
```

As you can see the `::T4CEP::CUSTOM::SERVICES::camstarDocumentSetService` is executed only if `[dict get $CamstarInputDat Transfer] eq "TRUE"`.

At last T4CEP executes the reverse mapping and writes the transaction status back into `description` attribute on the Item Revision using `::T4X::TC::MAPPING::storeReverseMappingAttribute`.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate Document to Document Set

To associate the previously created Documents T4CEP builds up the dictionary with a key and value pairs, which represent the relation to the Document Set.

```
set ObjectTag [string trimright [tpco_formatHEX16 $item] "A"]
set DocSetName ${fileName}_${ObjectTag}
dict set CamstarInputDat Properties Documents $item Properties Name \
  $DocSetName
dict set CamstarInputDat Properties Documents $item Properties Document \
  $DocSetName
dict set CamstarInputDat Properties Documents $item Properties
Description \
  $fileNameOriginal
```

Maintain Document Set fields

To learn about the maintained attributes take a look into the `TC_Object2EA_Object` by mapping for the `$TargetTypeName eq "DocumentSet"` in *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_object_mapping.sd*.

7. Part Transfer – Product

Create Product

Capability to create Product Revision in CEP based on Item Revision released from Teamcenter. The Product transaction uses the `CamstarInputDat` dictionary based on the well known T4x approach using preferences and mapping by calling `::T4X::TC::MAPPING::FieldMapping`. To learn more about mapping by invoking perform function take a look at the `performTransfer` in *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_object_mapping.sd*.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate Document Set

By default, if the ItemRevision in Teamcenter has a DataSet, T4CEP associates the previously created Document Set to the Part. It is done by adding the key and value into the `CamstarInputDat` as described below:

```
 dict set CamstarInputDat Properties DocumentSet \
  [::T4X::TC::MAPPING::FieldMapping $Item "item_id"]_\
[::T4X::TC::MAPPING::FieldMapping $ItemRev "item_revision_id"]
```

Associate Workflow

Capability to associate a previously created Workflow in CEP to a Product Revision. This association is useful during the manufacturing order dispatch process for container execution.

The knowledge about the Workflow exists during the 100 % transaction. To associate the Workflow to the Part works as follows:

```
dict set ProductDict Properties Workflow ${ItemName}_${ItemId}
```

The populating of the dictionary, shown in the code example above, is a part of the `PLMXML_Data2EA_Object_MapTopLevelProcessHeaderData` implementation.

Associate Bill of Process Override

Capability to associate a previously created Bill of Process Override in CEP to a Product Revision. This association is useful during the manufacturing order dispatch process to identify the variant / unit specific process content to be used during CEP Workflow execution.

As well as the Worfklow the knowledge about the Bill of Process Override is available in the 100 % transaction only. To associate the Bill of Process Override to the Part the following line are configured:

```
dict set ProductDict Properties BillOfProcess [dict get $BopFlowDict
Name]
```

The populating of the dictionary, shown in the code example above, is a part of the `performEATransfer` implementation.

Maintain Product fields

The maintained fields are the keys of the `ProductDict`. Note that the dictionary gets the key and value pairs if the variable `TransferOfBOM` equals `true` only. Go to the `PLMXML_Data2EA_Object_MapTopLevelProductHeaderData` procedure in *GS_ROOT//var/mmap/t4cep_mapping_config/t4cep_object_mapping.sd* to get more details.

8. Problem Report - Quality Event

Import Quality Event

Capability to create Problem Report in Teamcenter based on Quality Event released from CEP. The Problem Report transaction uses the `T4x Step Engine` framework based on the Railway approach. It creates a new Problem Report and adds the Items (for example Part, Operation) into the preconfigured folder of Problem Report. To learn more about mapping take a deeper look at the `MOMimportServicePR` in *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_custom_import_services.sd*:

```
set Steps {   \
            validateCall \
            parsePayloadPR \
            {ConnectTC gate} \
            createPR \
            getObjectId \
            addItemsToPR \
            {Response final} \
    }
```

9. Manufacturing BOP Transfer

Basic configuration

Bill of Process (BOP) transfer from Teamcenter is a complex transaction which results in create or update on multiple objects in CEP, not all of which exist as revisionable objects in the source system.

For example, the Camstar BOP references a series of Specification objects, which in turn refer to the Electronic Procedure & Task List objects. Both Electronic Procedure and Task Lists do not exist in Teamcenter and are derived from the Teamcenter BOP structure during the data transfer process by T4CEP and CIO.

Therefore, the sequence of calls to create or update the BOP is of key significance inside T4CEP mapping configuration files.

Following sections detail the CEP objects in the order of transfer from T4CEP interface.

> Caution:
>
> The OOTB T4CEP mapping configuration relies on the Classic Options & Variant functionality to configure the BOP for CEP.
>
> Before starting the workflow for the 100 % structure, please check if the Variant Rules in the BOP structures is configured. The name of the Variant Rule is used to build up the 100 % - specific objects; it is required and will be validated in mapping before to start the transferring.
>
> The use of Classic Options & Variant module is not mandatory; but used only as a template for OOTB setup.
>
> This can be easily modified by the solution deployment team based on customer specific requirements.

CEP ERP Route

Create ERP Route

Create ERP Route Step

Associate Change Package

Maintain ERP Route Step fields

Maintain ERP Route fields

Create ERP Route

Capability to create ERP Route in CEP based on the master plan definition in Teamcenter. The ERP Route is used to reference the milestone operations on the master plan, so that they can be matched with the ERP operation sequences for backflush, inventory updates etc.

It is the first part of the 150 % transfer transaction. If the workflow argument -ERPRouteSetup equals `true` and the `$TransferOfRouteList` as well; Use the `$TransferOfRouteList` to override the workflow argument value.

In a different way to other transfers, the ERP BOM transfer has a deviant behavior and consequently the specific implementation of the function `camstarStructureService` in *<GS_ROOT/var/mmap/t4cep_mapping_config/t4cep_custom_services.sd>*.

The dictionary `RouteListDict` contains steps. Each of the steps is stored in the `RoutesDict`.

Create ERP Route Step

Capability to create ERP Route Step in CEP based on the master plan structure definition in Teamcenter. The ERP Route Step is referenced on the ERP BOM so as to highlight steps of the CEP Workflow where Material Consumption needs to be recorded on the Shop Floor.

Firstly, the function `createRoute` ensures that ERP Route Step is created. Then the `addRouteToRouteList` attaches the previously created step to the ERP Route list.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the

pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Maintain ERP Route fields

The maintained fields are the keys of the `RouteListDict`:

```
dict set RouteListDict Name            ${ItemName}_${ItemId}
dict set RouteListDict Revision        $ItemRevision
dict set RouteListDict Description     $ItemName
dict set RouteListDict Notes $ItemDescription
dict set RouteListDict momPLMUID $Tag
```

Maintain ERP Route Step fields

The maintained fields are the keys of the `RoutesDict`:

```
dict set RoutesDict $RoutesDictIndex Properties Name ${HeaderName}_$
{HeaderId}
dict set RoutesDict $RoutesDictIndex Properties Revision $HeaderRevision
dict set RoutesDict $RoutesDictIndex Properties Description $HeaderName
dict set RoutesDict $RoutesDictIndex Properties ERPOperation \
  $HeaderSequenceNumber
dict set RoutesDict $RoutesDictIndex Properties momPLMUID $HeaderTag
```

CEP Spec

Create Spec

Capability to create specification (Spec) in CEP based on the master plan structure definition in Teamcenter. The Spec defines the activities to be carried out at a step in the master plan. The Spec is later referenced by both the Workflow and Bill of Process Override definitions in CEP to decide which set of scheduling and processing parameters are to be used during manufacturing order execution.

The corresponding object for the Spec is MEProcessRevision on the first level in the BOP. The Teamcenter BOP structure containing MEProcessRevision on the second level or deeper is not supported. During the executing of `PLMXML_Data2EA_Object_ProcessToDict` T4CEP analyze and validate the whole structure. In case if MEProcessRevision found, the `performEATransfer` will be not executed.

The transferring starts programmatically as a part of 100 % and 150 % transaction. See `transferCollection` for details.

CEP Specs must capture each variant configuration; this could be done with static Specs and a fixed Workflow.

T4CEP transfers the Workarea from the MEProcessRevision. Workarea has to exist as a first child element from the MEProcessRevision and represents the Workcenter in CEP and is an optional attribute. If you want to jump into more of the details, have a look at the call `[transferCollection SpecMaint $SpecsDict false]` in the *GS_ROOT/var/mmap/t4cep_mapping_config/t4cep_plmxml_object_mapping.sd*.

CIO API	Dict Name	is Structure
SpecMaint	SpecsDict	false

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Maintain Spec fields

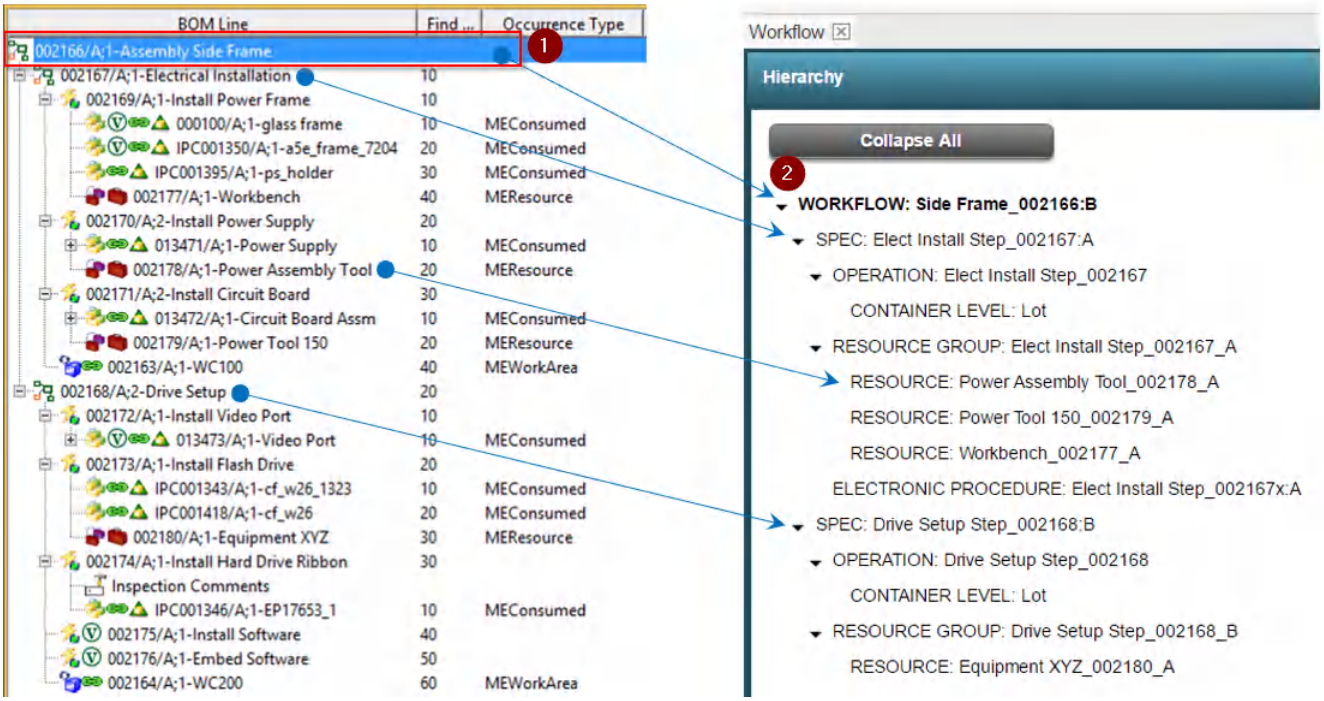
Capability to maintain CEP fields such as Workcenter, Description, and Notes on the Spec. The `$SpecsDict` stores the data from Teamcenter, which will be transformed into XML later on. The populating of `SpecsDict`dictionary begins with function call `createSpec`.

CEP Workflow

This is a para in the topic template.

Create Workflow

Capability to create a Workflow in CEP to represent the standard sequence of steps identified in the Teamcenter master plan definition to manufacture the product. The top level or header MEProcessRevision represents the Workflow. T4CEP transfers Workflow by posting to CIO API `WorkflowMaint`.



The 100 % and 150 % transactions use the same dictionary `BopFlowDict` as a basal data. The workflow agrument `–Transfer150PercentBOP` manages the transactions. If it equals `true`, then the dictionary key `TransferType` has the value `WorkflowMaint`.

```
if {$Transfer150PercentBOP eq "true"} {
          dict set BopFlowDict TransferType WorkflowMaint
                } else {
          dict set BopFlowDict TransferType BillOfProcessMaint
                }
```

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate ERP Route

Capability to associate a previously created ERP Route in CEP to Workflow definition.

The populating of dictionary `RouteListDict` happens by 100 % and by 150 % transaction. The transferring takes place by 150 % transaction. Per definition, the ERP Route has to exists or transferred to CEP before the Workflow.

```
if {$Status eq "OK" && $TransferOfRouteList eq "true"} {
  dict set RouteListDict Action Sync
  dict set RouteListDict TransferType Route
  addECOToDict RouteListDict
  set Status [::T4CEP::CUSTOM::SERVICES::camstarStructureService
```

```
$RouteListDict]
  if {[lindex $Status 0] eq "OK"} {
    set Status OK
  }
}
```

Please note that the variable `TransferOfRouteList` overrides the workflow argument – `ERPRouteSetup` value.

Associate Specs

Capability to associate previously created Spec(s) with corresponding Step(s) of the Workflow.

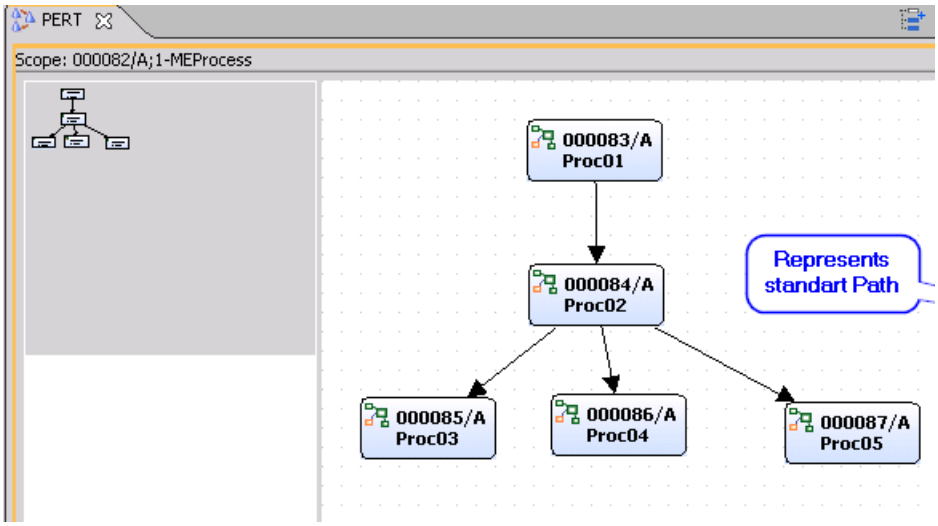
The function `addOverrideOrSpecToBopFlowDict` ensures that depending on the value of `Transfer150PercentBOP` the dictionary `BopFlowDict` gets the right key name `bopFlowType`; that is `Steps` in case of the WorkflowMaint transaction.

The configuration of Specs is applied in following order :

```
dict set BopFlowDict $bopFlowType $BopFlowDictSpecIndex Spec \
            [dict get $SpecsDict $BopFlowDictSpecIndex Name]...
```

Identify Standard Workflow Paths

Capability to identify the sequence of steps (also known as paths) to be performed on the CEP Workflow. This information is based on the PERT chart based definition of master plan sequence in Teamcenter. CEP Workflow differentiates between default and alternative paths, PERT chat supports default only. Currently, T4CEP sends the paths without to distinguish the type, CIO decides what is it the default path in workflow later on.

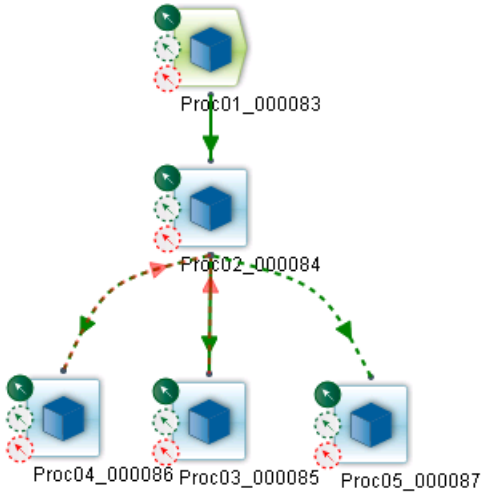


During the execution of `PLMXML_Data2EA_Object_ProcessToDict` T4CEP analyses the structure and creates, if any paths found, the `Paths` key and populates the dictionary with defined attributes.

```
dict set BopFlowDict Steps $Position Paths $PathsIndex Name $SucessorID
```

For the details on how to configure the Paths see `getPredecessors` in the *t4cep_plmxml_object_mapping.sd*.

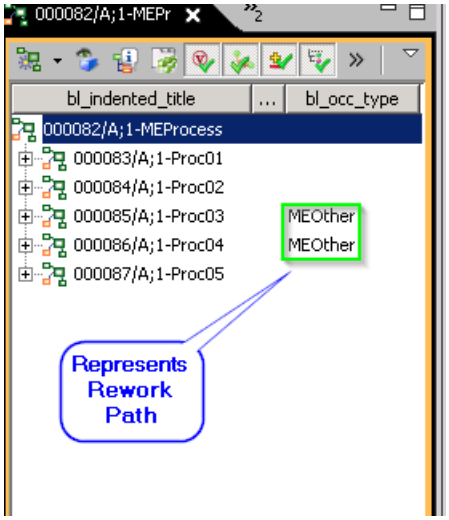
The Paths for the structure transfer example above looks as follows:



Identify Rework Workflow Paths

Capability to identify the steps to be setup as Rework type on the CEP Workflow. A Rework step is used as a manufacturing process to correct a problem or an undesirable result that occurred during manufacturing process execution.

This can be identified in Teamcenter against the MEProcess (CEP Step) definitions with an appropriate "Occurrence Type" assignment such as MEOther. Similar to the Standard Workflow Steps, the MEProcess identifying the Rework Step is also expected to be part of the PERT chart based definition of the master plan in Teamcenter



During the execution of `PLMXML_Data2EA_Object_ProcessToDict` T4CEP analyses the structure and creates, if any Rework steps found, the `ReworkPaths` key and populates the dictionary with defined attributes.

```
dict set BopFlowDict Steps $BopFlowDictSpecIndex ReworkPaths \
  $ReworkCounter Name ReworkName_${PredecessorID}
```

For the details on how to configure the Paths see `getPredecessors` in the *t4cep_plmxml_object_mapping.sd*.

Maintain Fields

Capability to maintain CEP fields such as Description, momPLMUID and Notes.

For the details on how to configure the attribute see `PLMXML_Data2EA_Object_MapTopLevelProcessHeaderData` in the *t4cep_plmxml_object_mapping.sd*. Look at the following example as a quick reference for the attribute mapping:

```
dict set BopFlowDict Description    $ItemName
dict set BopFlowDict Notes          $ItemDescription
dict set BopFlowDict momPLMID       $ItemId
dict set BopFlowDict momPLMRevision $ItemRevision
dict set BopFlowDict momPLMUID      $Tag
```

CEP Resource

Create Resource

Associate Change Package

Maintain Fields

Create Resource

Capability to create Resource in CEP, which identifies a machine, piece of equipment or any other non-material entity at particular physical location in the factory.

The transferring starts programmatically as a part of 100 % transaction. See `transferCollection` for details.

CIO API	Dict Name	is Structure
ResourceMaint	ResourcesDict	false

The populating of the dictionary `ResourcesDict` with Resources starts if the Occurance from type `MEResource` is found. Please note that the existence of Data Collection Definition (DCD) on the same Operation is not supported.

```
if {$OccurrenceSubType eq "MEResource" && $MEOPhasDCD eq "false" || \
   $OccurrenceSubType eq "METool" && $MEOPhasDCD eq "false"} {
  if {$Status eq "OK" && ! [dict exists $ResourcesDict $ResourceKey]} {
     # only if this resource has not been handled before!
    createResource $ResourceKey $ItemName $ItemDescription $Tag
  }
}
```

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the

pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Maintain Fields

Capability to maintain CEP fields such as Description, momPLMUID and Notes.

For the details on how to configure the attribute see `PLMXML_Data2EA_Object_MapTopLevelProcessHeaderData` in the *t4cep_plmxml_object_mapping.sd*. Look at the following example as a quick reference for the attribute mapping:

```
dict set BopFlowDict Description    $ItemName
dict set BopFlowDict Notes          $ItemDescription
dict set BopFlowDict momPLMID       $ItemId
dict set BopFlowDict momPLMRevision $ItemRevision
dict set BopFlowDict momPLMUID      $Tag
```

CEP Resource Group

Create Resource Group

Associate Change Package

Associate Resources

Maintain Resource Group Fields

Create Resource Group

Capability to create Resource Group in CEP, which is a collection of resources available on the shop floor to produce a particular variant / unit specific configuration of the product.

The transferring starts programmatically after `ResourceMaint` transferring as a part of 100 % transaction. See `transferCollection` for details.

CIO API	Dict Name	is Structure
`ResourceGroupMaint`	`ResourceGroupDict`	`true`

During the execution of `createResourceGroup` the initial structure in `ResourceGroupDict` will be created.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate Resources

After the function call `createResource` the just created Resource will be added into the `ResourceGroupDict` as an entry (see `addResourceToResourceGroup` for details).

```
dict set ResourceGroupDict $ResourceGroupIndex Entries \
 ${Name}_${ResourceKey} __name ${Name}_${ResourceKey}
```

Maintain Resource Group Fields

Please refers to the `createResourceGroup` function to learn more about the maintained fields.

CEP User-Defined Data Collection

Create User-Defined Data Collection

Associate Change Package

Maintain User-Defined Data Collection Fields

Maintain Data Point Fields

Create User-Defined Data Collection

Capability to create a User-Defined Data (DCD) Collection in CEP, which has the set of data points represented in Teamcenter as Data Collection Definition (DCD) object(s) in the MEOperation structure.

A DCD represents a point in the production process where data can or must be collected.

The transferring starts programmatically after `ResourceGroupMaint` transferring as a part of 100 % transaction. See `transferCollection` for details.

CIO API	Dict Name	is Structure
`UserDataCollectionDefMaint`	`DCDListDict`	`true`

The DCD can be identified in the XML structure by checking the Role on the particular level. If the Role has the name `Mes0MEDCD_Details`, we selected the revision has an DCD.

```
tplet Role $associatedAttachmentURIs getRole
if {$Role eq "Mes0MEDCD_Details"} { # TODO MAPPING COMING BELOW}
```

In the next step you can check the existence of the DCD list on the Operation and create a initial structure for the `DCDListDict`

```
if {$isDCDListAlreadyCreated eq "false"} {
  set Status [createDCDList $ItemId $ItemRevisionId $ItemName \
    $ItemDescription $Tag]
  set isDCDListAlreadyCreated "true"
}
```

Then create a DCD entry, also called DataPoint by calling `createDCD`

Finally, add the DCD created in the previous step into the `DataPoints` by executing `addDCDToDCDList`.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Maintain User-Defined Data Collection Fields

To create a DataPoint calls `the createDCD` function. Please have a look at this function to learn more about data mapping. The variable `DataType` stores information about the type of DCD extracted from the object on the MEOperationRevision having `associatedAttachment` with `Role` equals `Mes0MEDCD_Details`. T4CEP providing support for following types:

```
Mes0MEIntegerDCDForm

Mes0MEStringDCDForm

Mes0MEBooleanDCDForm
```

Please note that the DCD with type `Mes0MEBuyoffDCDForm` used to represent the e-Signature and TrainingsRequirement in CEP. To differentiate between objects use attribute `mes0ValuePlaceholder`. The value for e-Signature must be `eSig` and `TrainingsReq` for TrainingsRequirement accordingly.

```
if {$DataType eq "Mes0MEBuyoffDCDForm"} {
      addDCDtoTask $Name [dict get $dataValuesDict mes0ValuePlaceholder]
      return $Status
    }
```

In the `addDCDtoTask` the `Tasks` item gets the certain attribute and value.

```
if {$TrainingReqName eq "TrainingsReq"} {
      dict set TaskListDict $TaskListIndex Tasks $TaskIndex
TrainingReqGroup \
        $DCDName
  } elseif {$TrainingReqName eq "eSig"} {
      dict set TaskListDict $TaskListIndex Tasks $TaskIndex
ESigRequirement \
        $DCDName
    }
```

Those objects are not the part of the transfer and must exist in CEP before you start the transfer.

Maintain Data Point Fields

To learn which attributes are supported look at the `createDCDList` in *<GS_ROOT>/var/mmap/t4cep_mapping_config/t4cep_plmxml_object_mapping.sd*.

CEP Task List

Create Task List

Create Task (Instruction or Transaction type)

Associate Document Set

Associate E-Signature Group to Task

Associate Training Requirement Group to Task

Associate User-Defined Data Collection to Task

Associate Resource Group to Task List

Associate Change Package

Maintain Task List Fields

Maintain Task Fields

Create Task List

Capability to create a Task List in CEP, which represents the group of Tasks to be performed on a container at a manufacturing workflow step to build a particular variant / unit configuration of the product.

The transferring starts programmatically after `ResourceGroupMaint` and `UserDataCollectionDefMaint` transferring as a part of 100 % transaction. See `transferCollection` for details.

CIO API	Dict Name	is Structure
`TaskListMaint`	`TaskListDict`	`true`

Please note that there are some limitations on the structure in Teamcenter, which will cause the abort already in the mapping evaluation step.

- The attribute `FindNr` of MEOperations must be unique under the same MEProcess.
- MEOperation cannot has both DCD and MEConsumed assignments.

Create Task (Instruction or Transaction type)

Capability to create a Task in CEP, which is a unit of work required to be completed by the Shop Floor operator for the container to move along the manufacturing process.

Based on the MEOperation content found in Teamcenter, the Task Type can be either:

- Instruction – default type. Typically used for identifying Data Collection Definitions in CEP.
- Transaction – If one or more MEConsumed assignments are found under the MEOperation. Transaction Type tasks are not allowed to have Data Collection Definitions in CEP. The Task type must be a Transaction Task with default TransactionPage = ComponentIssue_VPR2.

```
dict set TaskListDict $TaskListIndex Tasks $TaskIndex \
 TransactionPage ComponentIssue_VPR2
```

Please refer to the function `addTaskTypetoTask` for more details about how to handle the type on the task.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate Document Set

Capability to associate a previously created Document Set in CEP to a Task. This Document Set is expected to contain references to the Textual Work Instruction and/or 3D Snapshot representation in JPG format and/or EWI URL Documents setup in CEP as part of Document transfers.

If at least one `memberURI` (file) exists on the `AssociatedDataSet`, the function `checkDocExistence` returns the `DocumentSet` with a value of the parent object. The name is the concatenated string from `<item_id>`, `_`, and `<itemrevision_id>`.

To associate the value of the `DocumentSet` populate the dictionary with a key `DocumentSet` and the value from `$DocumentSet`.

Associate E-Signature Group to Task

Capability to associate an existing E-Signature Group in CEP to a Task. E-Signature Group is used to represent one or more electronic signature requirements which need to be performed for the task to complete.

If the DCD in Teamcenter has a type `Mes0MEBuyoffDCDForm` and the attribute `mes0ValuePlaceholder` has the value `eSig`, the function `addDCDtoTask` creates the key with a name `ESigRequirement` and value from the `$DCDName` with a DCD's name.

```
dict set TaskListDict $TaskListIndex Tasks $TaskIndex \
  ESigRequirement $DCDName
```

Associate Training Requirement Group to Task

Capability to associate an existing Training Requirement Group in CEP to a Task. Training Requirement Group is used to identify one or more individual training requirements which the operator needs to be have fulfilled to perform a task.

If the DCD in Teamcenter has a type `Mes0MEBuyoffDCDForm` and the attribute `mes0ValuePlaceholder` has the value `TrainingsReq`, the function `addDCDtoTask` creates the key with a name `TrainingReqGroup` and value from the `$DCDName` with a DCD's name.

```
dict set TaskListDict $TaskListIndex Tasks $TaskIndex \
  TrainingReqGroup $DCDName
```

Associate User-Defined Data Collection to Task

If the DCD in Teamcenter has a type `Mes0MEBuyoffDCDForm` and the attribute `mes0ValuePlaceholder` is not `TrainingsReq` or `eSig`.

Or if the DCD in Teamcenter has one of the supported types see **Maintain User-Defined Data Collection Fields**, the function `addDCDtoTask` creates the key with a name `DataCollectionDef` and value from the `$DCDName` with a DCD's name. Additionally `InstructionType` with value `2` is required.

```
dict set TaskListDict $TaskListIndex Tasks $TaskIndex \
  DataCollectionDef $DCDName
      dict set TaskListDict $TaskListIndex Tasks $TaskIndex \
  InstructionType 2
```

Associate Resource Group to Task List

To associate the particular Resource Group to the Task List call the `addResourceGroupToTaskList`. It sets the key `WorkStationGroup` with a value from the same MEOperation `[dict get $ResourceGroupDict $ResourceGroupIndex Name]` as shown in the following example:

```
dict set TaskListDict $TaskListIndex WorkStationGroup \
 [dict get $ResourceGroupDict $ResourceGroupIndex Name]
```

Maintain Task Fields

Please refer to the function `createTask` to learn how to configure the attribute's mapping and which fields are maintained by default.

Maintain Task List Fields

Please refer to the function `createTaskList` to learn how to configure the attribute's mapping and which fields are maintained by default.

CEP Bill of Process Override

Create Bill of Process Override

Associate Change Package

Associate E-Procedure

Associate Spec

Maintain Fields

Create Bill of Process Override

Capability to create a Bill of Process (BOP) Override in CEP to capture a variant / unit specific manufacturing process content required to build a unique configuration of the product. From

Teamcenter perspective, the Bill of Process Override represents the process plan content configured to a unique configuration (Variant Rule/Effectivity) of the Product.

During order execution, the container on the shop floor goes through the standard sequence of steps identified in the CEP Workflow and overlays it with the Bill of Process Override content to manufacture the unique variant / unit configuration of the product.

T4CEP transfers BOP Override by posting to the CIO API `BillOfProcessMaint`

The 100 % and 150 % transactions use the same dictionary `BopFlowDict` as a basis. The workflow agrument `–Transfer150PercentBOP` manages the transactions. If it equals `false`, then the dictionary key `TransferType` has the value `BillOfProcessMaint`.

```
if {$Transfer150PercentBOP eq "true"} {
  dict set BopFlowDict TransferType WorkflowMaint
} else {
  dict set BopFlowDict TransferType BillOfProcessMaint
}
```

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate E-Procedure

Capability to associate auto-created variant / unit specific E-Procedure to the Bill of Process step

By default, E-Procedure Name mapped to the Task List Name as shown in the following example:

```
if {[string toupper [string index $Transfer150PercentBOP 0]] eq "F"} {
 dict set BopFlowDict $bopFlowType $BopFlowDictSpecIndex
ElectronicProcedure \
   [dict get $TaskListDict $TaskListIndex Name]
}
```

As you can see it happens during the 100 % transaction, that means that the Task List name must contain a variant-specific part. The name configuration is done by overriding of key values:

```
set EProcedureName [dict get $BopFlowDict BillOfProcessOverrides \
  $BopFlowDictSpecIndex ElectronicProcedure]
append EProcedureName "_${VariantRuleNameBOP}"
dict set BopFlowDict BillOfProcessOverrides \
  $BopFlowDictSpecIndex ElectronicProcedure $EProcedureName
```

Associate Spec

The function `addOverrideOrSpecToBopFlowDict` ensures that depending on the value of `Transfer150PercentBOP` the dictionary `BopFlowDict` gets the right key name `bopFlowType`; that is `BillOfProcessOverrides` in case of the BillOfProcessMaint transaction.

```
 if {[string toupper [string index $Transfer150PercentBOP 0]] eq "F"} {
      set bopFlowType "BillOfProcessOverrides"
    }
```

The configuration of Specs is applied in following order :

```
dict set BopFlowDict $bopFlowType $BopFlowDictSpecIndex Spec \
            [dict get $SpecsDict $BopFlowDictSpecIndex Name]...
```

Maintain Fields

Capability to maintain CEP fields such as Description, momPLMUID and Notes.

For the details on how to configure the attribute see `PLMXML_Data2EA_Object_MapTopLevelProcessHeaderData` in the *t4cep_plmxml_object_mapping.sd*. Look at the following example as a quick reference for the attribute mapping:

```
dict set BopFlowDict Description    $ItemName
dict set BopFlowDict Notes          $ItemDescription
dict set BopFlowDict momPLMID       $ItemId
dict set BopFlowDict momPLMRevision $ItemRevision
dict set BopFlowDict momPLMUID      $Tag
```

CEP ERP BOM

Create ERP BOM

Capability to create ERP BOM in CEP, which is used to identify the list of Materials required to assemble a particular variant / unit configuration of the product on the shop floor.

From Teamcenter perspective, this is a collection of MEConsumed assignments found under specific variant / unit configured process plan.

The transferring starts in code sequence after the successful variant-specific Part transfer as a part of 100 % transaction if the workflow argument -ERPBOMSetup equals `true` and the `$TransferOfBOM` as well; the `$TransferOfBOM` overrides the workflow argument value.

In a different way to other transfers, the ERP BOM transfer has a deviant behavior and consequently the specific implementation of the function `camstarMBOMService` in *<GS_ROOT/var/mmap/t4cep_mapping_config/t4cep_custom_services.sd>*.

Associate Change Package

Capability to associate the object to a specific Change Package in CEP based on ECN released from Teamcenter. All objects will be transferred in context of ECN. It means that by default you have to put:

- all Itemrevisions, MEProcessRevisions and MEOperationRevisions into Solution folder,
- the unconfigured CC object into the Reference folder by 150 % transaction,
- and the configured CC object into the Solution folder by 100 % transaction.

The XML payload contains the information about the associated ECN; this information will be processed during the sync transaction in CIO to CEP.

The variable `ChangePackageToAssign` stores the `item_id` and `item_revision_id` attributes from the ChangeNoticeRevision. The slash "/" is used to divide two attributes if CEP sends the approval message back to T4CEP.

Teamcenter out of the box data model has certain restrictions and conditions regarding inserting of particular objects into the Solution or Reference folder. The default configuration presupposed that the pseudo folder mentioned above have no restrictions and can be populated with all object, which used for transferring in examples.

Associate ERP Route

To associate the ERP Route to the ERP BOM done by adding the `ERPRoute` and `ERPRouteRevision` as shown in the following lines:

```
# Fill the header information of the bom dict
      dict set BOMDict ERPRoute ${ItemName}_${ItemId}
      dict set BOMDict ERPRouteRevision $ItemRevision
```

Associate ERP Route Step

During the execution of mapping by calling the function `addMaterialToMaterialList` the ERP Route Steps added to the `MaterialList` as shown below:

```
dict set BOMDict Properties MaterialList $BOMPositionIndex RouteStep
$RouteStep
```

Maintain ERP BOM Fields

Take a look at the function `PLMXML_Data2EA_Object_MapTopLevelProductHeaderData` to learn which fields are maintained.

A. 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

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.

FN4S
Closed Loop Manufacturing for SAP S/4HANA®

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

1 JSON.org

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.

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

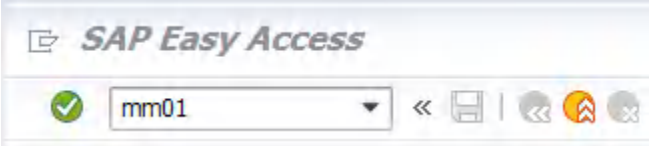
TCL
is a high-level, general-purpose, interpreted, dynamic programming language.

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 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.