Contents

Preface 5

Introduction 1-1

Overview 2-1

Installation
Basic Installation 3-1
Installation of an OFBiz server 3-2
Installation of the Oracle demo schema OE 3-3
Installation of the Demo ERP WSDL and demo service 3-4
Demo JDBC Installation 3-7

Mapping Files
Necessary Modifications 4-1
Files and Procedures 4-2
Overview 4-2
t4ea_mapping_config.sd 4-3
t4ea_object_mapping.demo.sd 4-3
t4ea_genbom_mapping.demo.sd 4-4
t4ea_custom_services_demoerp.sd 4-5
t4ea_custom_services_ofbiz.sd 4-6
t4ea_custom_functions_oracle.sd 4-6
t4ea_oracle_connection_handling.sd 4-7
t4ea_item_import_ofbiz.sd 4-7
t4ea_item_import_oracle.sd 4-9
t4ea_bom_item_import_template.sd 4-10
t4ea_query_mapping.demo.sd 4-11
t4ea_prop_mapping.demo.sd 4-12
t4ea_idgen_mapping_template.sd 4-14
t4ea_connection_mapping_template.sd 4-14
t4ea_async_services.sd 4-14
t4ea_meta_import_template.sd 4-15
t4x_mapping_config.sd 4-15
t4x_custom_mapping_toolbox.sd 4-16
t4x_user_exit_template.sd 4-16
t4x_workflow_batchjob_template.sd 4-16

How to use the demo scenario
Connection Handling 5-1
Transfer to OFBiz 5-1
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: July 2018

Legal notice:

All rights reserved. No part of this documentation may be copied by any means or made available to entities or persons other than employees of the licensee of the Teamcenter Gateway or those that have a legitimate right to use this documentation as part of their assignment on behalf of the licensee to enable or support usage of the software for use within the boundaries of the license agreement.

© 2011-2018 Siemens Product Lifecycle Management Software Inc.

Trademark notice:

Siemens, the Siemens logo and SIMATIC IT are registered trademarks of Siemens AG.

Camstar and Teamcenter are trademarks or registered trademarks of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries.

Oracle is a registered trademark of Oracle Corporation.

SAP, R/3, SAP S/4HANA®, SAP Business Suite® and mySAP are trademarks or registered trademarks of SAP or its affiliates in Germany and other countries.

TESIS is a registered trademark of TESIS GmbH.

All other trademarks, registered trademarks or service marks belong to their respective holders.
1. Introduction

The Teamcenter Gateway for Enterprise Applications (T4EA) software solution is a general purpose integration software that provides data and process integration between Teamcenter by Siemens PLM Software and enterprise applications (EA) such as ERP or other database enterprise systems, including legacy applications.

Note:

- The demo scenarios and implementations mentioned in this manual are provided to demonstrate the capabilities of T4EA only. Neither is the selection of EA systems representative, nor is T4EA restricted to the scenarios presented here.

- It is not necessary to have access to or to install the demo Enterprise Applications mentioned in this guide to set up a working T4EA installation.

- For demo purposes, several of the installations mentioned in this manual can be done on a single host together with Teamcenter. This is not recommended for productive or test environments!

- The demo mappings and configurations are not intended for productive use and are not optimized in any way since clarity (which is the primary aim of the demos) often contradicts to optimization.

- The T4EA demo template (BMIDE template) should not be applied to a Teamcenter installation that is intended to be used for production or test, since this may unnecessarily inhibit applying any future templates.
2. Overview

T4EA provides a demo template to show how integrations with Enterprise Applications can be implemented. The template requires the standard “out of the box” data model in Teamcenter and the T4EA base template (as described in the Teamcenter Gateway - Installation Guide) and integrates Teamcenter with following Enterprise Applications:

- The open source ERP system Apache OFBiz. This system was chosen because it is freely available, easy to install and has interfaces for a basic integration. The integration is shown here only for demonstration purposes and should not be understood as a recommendation for any specific ERP system.

- A demo table in an Oracle database (demo schema OE, table product_information).

- A fictional Demo ERP system by interfacing with a demo SOAP web service frontend with no real ERP implementation. This demo ERP is presented to propose some useful web service interfaces and to provide an appropriate, ready-to-use gateway configuration.

- A file based interface is used to import BOMs to Teamcenter.

There are a lot of possible business processes and integration scenarios. The sample templates implement four typical use cases that show different T4EA functionalities:
• Search an item in the Enterprise Application and display the results as external data in the Teamcenter GUI for saved queries (T4EA external query). The results can be imported to Teamcenter if needed (see next item) and the resulting object can be used within Teamcenter.

• Import data from the search result (last item) or from other external sources to Teamcenter (T4EA mapping and import).

• Choose an Item revision and transfer the revision, an attached document or the BOM structure to the Enterprise Application using a Teamcenter workflow (T4EA mapping and transfer).

• Display properties of the single item or the product structure in the external Enterprise Application for the related Teamcenter object (T4EA data view)

The availability and the implementation of these use cases depend on the capabilities and interfaces of the Enterprise Application(s). Other use cases can be implemented in a similar way. These include transfers or imports of different source and target types (different item types, BOMs, datasets).

You can use the demo scenario with only one Enterprise Application available or with several or all systems.
3. Installation

3.1 Basic Installation

You have to install T4EA as described in the Teamcenter Gateway - Installation Guide, including the T4EA demo template to be able to use the demo scenarios.

You also need to install the CXF libraries and the Oracle database driver.

Some actions in the demo scenario require a T4EA job client to be running. So please select “Configuration/Job Agent”, click the “plus” icon on the upper right and add a job agent with “Execute all jobs” to your T4EA Gateway Service:

![Job Agent Configuration](image)

Then please press the green check to apply your changes. T4EA will ask you to restart the GS, which you should do now.

Additionally you have to have access to the applications used as the demo Enterprise Applications (OFBiz and Oracle schema OE).
3.2 Installation of an OFBiz server

Note:

Although OFBiz offers SOAP web services, the services and the implementation of their client bindings in the T4EA demo scenario are not representative, because OFBiz services use the outdated “RPC/encoded” style. This style is supported by T4EA (hence the demo scenario works) but up-to-date SOAP services should use the modern “document/literal” style, which is supported much better in T4EA. Please see Teamcenter Gateway - Connectivity Guide for details.

You can find instructions on how to install and use “The Apache Open For Business Project” (Apache OFBiz) on the OFBiz web site: http://ofbiz.apache.org. Download the software and install, following the instructions on the link labeled “Apache OFBiz Getting Started” and then “Demo and Test Setup Guide” (https://cwiki.apache.org/confluence/display/OFBIZ/Demo+and+Test+Setup+Guide). It is sufficient to use the packaged Derby database and the included Tomcat application server. Be sure to populate the database with some helpful demo data using the “run-install” Ant-Target as described in the “Demo and Test Setup Guide”.

Enabling necessary web services

OFBiz offers a bunch of web services in the out-of-the-box configuration. For the demo scenario, you need to enable some more web services. The following instructions apply to version 12.04 and 13.07 of OFBiz. All paths are relative to the installation directory. Please add the attribute export="true" to the “service” XML tags in the files listed in the following table (or change the value from “false” to “true” if already present). Example: in applications/product/servicedef/services_view.xml change the line <service name="getProduct" engine="java" to <service export="true" name="getProduct" engine="java".

List of files where to add the export="true" to service XML tags:

<table>
<thead>
<tr>
<th>Service name</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>echoService</td>
<td>Test connection</td>
<td>framework/common/servicedef/services.xml</td>
</tr>
<tr>
<td>createProduct</td>
<td>Create a product</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>updateProduct</td>
<td>Update a product</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>getProduct</td>
<td>Get product details</td>
<td>applications/product/servicedef/services_view.xml</td>
</tr>
<tr>
<td>createProductAssoc</td>
<td>Create BOMline</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>updateProductAssoc</td>
<td>Update BOMline</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>deleteProductAssoc</td>
<td>DeleteBOMline</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>Service name</td>
<td>Description</td>
<td>Location</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>getAssociatedProducts</td>
<td>Get BOM structure</td>
<td>applications/product/servicedef/services_view.xml</td>
</tr>
<tr>
<td>findProductById</td>
<td>Get product</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>performFindList</td>
<td>General search</td>
<td>framework/common/servicedef/services.xml</td>
</tr>
<tr>
<td>createProductContent</td>
<td>Create content for product</td>
<td>applications/product/servicedef/services.xml</td>
</tr>
<tr>
<td>createContent</td>
<td>Create document content</td>
<td>applications/content/servicedef/services.xml</td>
</tr>
<tr>
<td>createDataResource</td>
<td>Associate a resource with a product</td>
<td>applications/content/servicedef/services_data.xml</td>
</tr>
<tr>
<td>createFixedAsset</td>
<td>Create a Fixed Asset</td>
<td>applications/accounting/servicedef/services_fixedasset.xml</td>
</tr>
<tr>
<td>updateFixedAsset</td>
<td>Update a Fixed Asset</td>
<td>applications/accounting/servicedef/services_fixedasset.xml</td>
</tr>
</tbody>
</table>

You can review the available services using the OFBiz Web interface interactively. Use this URL: https://<yourofbizhost>:8443/webtools/control/ServiceList. This will list all services. Clicking on one of them will show whether it is exported or not.

### Changing the OFBiz service port

Since in a demo scenario OFBiz might be installed on a host where Teamcenter is also installed, the demo assumes a port for OFBiz that is different from the default. To change the OFBiz port to e.g. 8081:

2. Change all ports from 8080 to 8081 in `framework/service/config/serviceengine.xml` and in `framework/catalina/ofbiz-component.xml`.

### 3.3 Installation of the Oracle demo schema OE

You can find installation instructions on the Oracle web page when searching for “Installation Oracle demo schema OE”, for example using this link: [http://docs.oracle.com/cd/B28359_01/server.111/b28328/installation.htm#i6806](http://docs.oracle.com/cd/B28359_01/server.111/b28328/installation.htm#i6806).

On a Teamcenter server with an Oracle database installation of these schemas does not functionally interfere with the Teamcenter database.

T4EA has no JDBC driver bundled, so you first have to install a JDBC driver matching your database for the demo. See [JDBC Installation](#) about details on installing JDBC drivers.
3.4 Installation of the Demo ERP WSDL and demo service

T4EA comes with a proposal for a web service as it could be implemented by an ERP system. This proposal includes operations for creation and update of articles, documents and product structures (BillOfMaterials) and operations to get the data of a given article, the document data of an article and the (single-level) product structure of a head-article. The operations contain only a minimum of attributes and have to be extended for real data.

The proposal comes in the format of a WSDL (web service description language). The T4EA demo scenario is configured to work with and access a web service conforming to this WSDL. To use the demo ERP in the T4EA demo scenario, you must add an implementation to this WSDL, as it is only an interface description, not a real interface. An easy way to do so is using SoapUI, a free and powerful tool for testing SOAP web services. There are other tools with similar capabilities, e.g. Eclipse extensions or several commercial tools. For simplicity this explanation will refer to SoapUI only.

SoapUI can add a dummy implementation to a given WSDL. See [http://www.soapui.org](http://www.soapui.org) for downloading SoapUI and details of the usage.

Here are the steps you have to do:

- Download and install SoapUI (the free version is sufficient, “pro” is not necessary, LoadUI and the HermesVM is not required).

- Add a new SoapUI project:
• In the “Initial WSDL/WADL” field, enter the path to the WSDL in the T4EA installation directory which has the path: `var/template/t4eademo/T4EADemoERP.wsdl`. Please also activate the check box “Create Requests”.

• Generate all Mock Services with right mouse button:
• On the next screen of the wizard, make sure all operations are activated. Tick the “Start MockService” check box. Check the port and remember it:

• Click OK and choose a name for the service. The demo configuration assumes the name “T4EADemoERPBinding”: 
• Now edit the responses according to your needs with some demo values. Especially edit the Echo response “Code” value to “OK:

• The mock service is now ready to respond to requests from T4EA.

3.5 Demo JDBC Installation

For a demo installation you can use any database product where a JDBC driver is available. T4EA does not contain any JDBC or ODBC driver. The demo scenario uses the Oracle database, so you first have to install the Oracle JDBC driver. Please see Teamcenter Gateway - Installation Guide for details.

• Download MYSQL from http://www.mysql.com/downloads. Installing the product on Windows may require updating the .NET platform and the C runtime libraries.
• Choose to install the “Developer Default” on a “Developer Machine”. Choose a MySQL root password you can remember (e.g. “geheim”).

• Copy the JDBC driver mysql-connector-java-5.1.15-bin.jar from the MySQL installation (directory C:\Program Files\MySQL\MySQL Connector J/) to the lib/modules-directory of the T4EA installation.

The MySQL installation installs several demo databases. T4EA contains a demo script jdbc_connector_demo_mysql.tcl that accesses one of these demo databases named “world”. You can run this script from the T4EA Gateway Service Admin GUI choosing the “Script/Scripts” menu entry and then clicking “Demonstrate JDBC Connector via MySQL”. This script will essentially execute the following TCL code sequence:

```tcl
# Import the ::T4X::OBJECTS::* namespace
namespace import ::T4X::OBJECTS::*

# Load the JDBC module
set Module [tpmodule de.thesis.plmware.objects.module.jdbc com.mysql.jdbc.Driver]
tpwith $Module setLogLevel ALL

set username "root"
set password "geheim"
set dbServerHost "localhost"
set database world

set connUrl "jdbc:mysql://${dbServerHost}/${database}"
tplet Conn $Module getConnection $connUrl $username $password

# Sending an SQL update query to the DBMS
tpplet statement $Conn prepareStatement "update country set HeadOfState=? where code=?"
tpwith $statement setString 1 "Joachim Gauck"
tpwith $statement setString 2 "DEU"
tplet Result $statement executeUpdate

# Sending an SQL query to the DBMS
tpplet statement $Conn prepareStatement "select * from country where name like ?"
tpwith $statement setString 1 "ger%"
tplet CollectionsResultSet $statement executeQuery

# Retrieve some meta data (optional!)
tplet metaData $CollectionsResultSet getMetaData
tplet columnCount $metaData getColumnCount
array set columnNames {}
for {set i 1} {$i <= $columnCount} {incr i} {
    set columnNames($i) [tpwith $metaData getColumnName $i]
}

# Retrieve data from the ResultSet
while {[tpwith $CollectionsResultSet next]} {
    tplet Name $CollectionsResultSet getString Name
    tplet LocalName $CollectionsResultSet getString LocalName
    puts "Name: $Name"
    puts "LocalName: $LocalName"
}

# close the result set and the statement
```
# - this is important to free resources in the JDBC adapter!
tpwith $CollectionsResultSet close
unset CollectionsResultSet
tpwith $statement close
unset statement
4. Mapping Files

4.1 Necessary Modifications

The files in the subdirectory var/mmap/t4ea_mapping_config of the T4EA installation directory usually should be edited and renamed heavily to fit the customer’s needs. In a demo installation, only slight modifications concerning the server names and ports are necessary. Especially you need to modify some settings in the file var/mmap/t4ea_mapping_config/ t4ea_mapping_config.rfdt in the T4EA installation directory. Please review them and adapt them to your local system requirements:

```bash
# Teamcenter connection parameter
set Status [::ITK::setConnectionParameters "user" "password" "Engineering"]

# --- SOAP Webservice ---
# ERP server connection (parameters: system, URL, username, password)
::T4EA::CONNECTION2EA::setConnectionInfoPlain DemoERP1 \
  "http://<yourERPServer>:8089/T4EADemoERPBinding" "" ""
# Share for document transfer
set ::ERPTransferArea                               "C:/Temp/ERPTransfer"
# Flag for writing xml file to transfer directory ::ERPTransferArea (0/1)
set ::WriteTransferFile2ERPTransferArea             0

# --- Apache OFBiz --
# OFBiz server connection (parameters: system, URL, username, password)
::T4EA::CONNECTION2EA::setConnectionInfoPlain OFBiz1 \
  "http://<yourERPServer>:8080/webtools/control/SOAPService" admin ofbiz

# --- JDBC connections to Oracle databases --
# (parameters: system, jdbc:oracle:thin:<hostname>:<port>:<databasename>,
# <user>, <password>)
::T4EA::CONNECTION2EA::setConnectionInfoPlain Oracle1 \
  "jdbc:oracle:thin:@<yourOracleServer>:1521:<db1>" OE OE

# this only sets the connection string, but no credentials
# (cannot be used for batch processing)
::T4EA::CONNECTION2EA::setConnectString Oracle2 \
  "jdbc:oracle:thin:@<yourOracleServer>:1521:<db2>"
```

Failure to adapt these setting will result in different error messages while executing the demo scenario. Remember to rebuild the t4ea_mapping_config.rfdt as described in the Teamcenter Gateway - Installation Guide.

Note:
The T4EA_Async_Article_transfer workflow is a sample configuration that does not execute the actual transfer in the user’s session but instead passes the transfer task to the T4EA job server. For this sample to work, you must create the “t4xbatch” user in the dba group and configure that user in t4ea_mapping_config.sd.
4.2 Files and Procedures

4.2.1 Overview

In the T4EA installation directory, the subdirectory var/mmap/t4ea_mapping_config contains the following T4EA mapping files:

Additionally some files common to all T4x products are in var/mmap/t4x_mapping_config:

There may be more files in the directories or more directories in var/mmap in case your installation package is also able to support other T4x products, e.g. T4S installation.
The following chapters give an introduction to the content of the files and the semantics of the procedures. However to fully understand the details of each mapping, you will need to carefully read the TCL code and the comments included. Having understood the demo mappings, you will be able to implement your own mapping based on the samples.

### 4.2.2 t4ea_mapping_config.sd

The “Mapping Configuration” is the central mapping file. It determines which other mapping files should be active (by including the other files with the TCL `source` command), contains some global variable settings for server names, ports, connection parameters and credentials. For a demo installation this is the only file which needs to be adapted. When transporting a T4EA configuration from one installation to another (e.g. from test to production) only this file should have to be adapted.

`t4ea_mapping_config.sd` is a fixed name that must be present in the `var/mmap/t4ea_mapping_config` directory. The names of the other mapping files (but not the namespaces within the files) may be changed as needed.

### 4.2.3 t4ea_object_mapping_demo.sd

This mapping file contains the configuration for the transfer of item revisions from Teamcenter to the Enterprise Applications.

**TC_Object2EA_Object**: This procedure maps Teamcenter attributes to EA attributes (respectively their representation in the used services). Teamcenter attributes can be read using the procedure `::T4X::TC::MAPPING::FieldMapping`, which retrieves values from an internal data structure, resulting from the previous extraction from the Teamcenter database. This procedure uses a kind of path notation to specify the storage location of the read attributes. This allows reading item and revision attributes as well as master form attributes and values of other objects related to the revision (which is the initial target of the transfer). Note that at the time of execution of the procedure `TC_Object2EA_Object`, the TCL interpreter does not need to have a Teamcenter connection. The mapped keys and values are stored in a TCL dictionary (named `ERPInputDat`), that holds key-value pairs where the keys are attribute names of the service data model. Again at the time of execution there is no connection to the EA. Instead a later step (the procedure `performTransfer`) will do the actual transfer.

The demo mapping for non-structured objects knows six kinds of transfers: “Article”, “Document”, “ProductInformation”, “Product”, “ProductDocument” and “FixedAsset”. These transfers are distinguished by the parameter `TargetTypeName`. The implementation of the mapping and the internal structure of the `ERPInputDat` dictionary are different depending on the `TargetTypeName`.

**performTransfer**: is the implementation of the actual transfer of the previously prepared data. The values to be transferred are available in the TCL dictionary `ERPInputDat`. Again this function distinguishes between different values for `TargetTypeName`. For transferring the data, the procedure uses the target-system specific procedures

- `::T4EA::CUSTOM::SERVICES::DemoERPService` for Demo ERP
These procedures are implemented in the mapping files starting with t4ea_custom_.

getAddressInfo: gets called after the transfer has completed to retrieve all available data from the external object. This data is potentially necessary for the "reverse mapping", i.e. write attributes of the external system back to Teamcenter. For the demo scenarios this step is not necessary so the procedure does nothing and returns OK in every case.

EA_Object2TC_Object: This procedure maps attributes of the external system back to the attributes of the Teamcenter object (without actually writing to Teamcenter – this is handled by the calling procedure later on). The most prominent example of such data is the ID of the external object. This ID usually has to be stored in Teamcenter to be able to relate the Teamcenter object to the new external object in future transfers (Note that this is not necessary if your external application uses the Teamcenter ID, see below). Often the status of the transfer (Success or Failure) or the date of this transfer is also mapped to some Teamcenter attributes. The procedure call : :T4X::TC::MAPPING::storeReverseMappingAttribute can be used to store mapped values that should be written to the Teamcenter object later on. For the Demo ERP scenario storing the external ID in Teamcenter is not necessary, because the item ID is assumed to be identical to the ID in the Demo ERP system.

callConnection2EA4Transaction: This is an optional procedure and can be used to tailor the usage of an EA connection depending on e.g. the transaction ID or the TargetTypeName. In the simplest case it just delegates the decision to : :T4EA::CONNECTION2EA::CUSTOM::MAPPING::checkConnection2EA4Transaction.

callCustomerRuleHandler: may be used to check some conditions in the Teamcenter workflow before the actual transfer handler gets executed. It is not used in this context.

4.2.4 t4ea_genbom_mapping_demo.sd

This mapping file implements the transfer of a Teamcenter BOM to the Demo ERP. Note that each procedure has the parameter TargetTypeName. The demo scenario supports BOM transfer to the Demo ERP (TargetTypeName “BillOfMaterial”, using structure based on Teamcenter bom view, and TargetTypeName “BOM4Relation”, using structure based on Teamcenter relations) and OFBiz (TargetTypeName “ProductAssociation”, using structure based on Teamcenter bom view).

getTcData: This procedure must retrieve the Teamcenter BOM information. Usually this is done by simply calling : :ITK::getObjectData passing all parameters unchanged and adding the TargetTypeName for the BOMlines of the transfer. Using this mechanism, you can define different BOMline TargetTypeNames for different BOM TargetTypeNames.

TC_Object2EA_ObjectPosition: This procedure maps Teamcenter data to a single BOM line of the Enterprise Application. Use ::T4X::TC::MAPPING::IndexedFieldMapping to retrieve data for the
current Teamcenter BOM line and the related item and revision. The procedure does not transfer any
data to the Enterprise Application, this happens later in the performTransfer procedure. Instead the
mapped data must be stored in the ERPInputDat dictionary for later use. Note that
TcPositionIndex is a parameter to the procedure which is only a consecutive number of the
BOMline, not its sequence number!

**TC_Object2EA_Object**: This procedure maps the attributes of the BOM header only. Again this
procedure only stores the mapping results in ERPInputDat and postpones the real transfer to
performTransfer.

**EA_Object2TC_Object**: As for the object transfer, this procedure can be used to write some Teamcenter
attributes as a result of the transfer. Here only the comment field on the master form is updated.

**getObjectInfo**: This procedure can be used to retrieve data from the Enterprise Application, e.g. to be
able to later compare the Enterprise Application BOM to the Teamcenter BOM. For the demo scenario
this procedure returns OK only.

**performTransfer**: Implements the real transfer of the previously mapped data to the Enterprise
Application.

**callCustomerRuleHandler**: may be used to check some conditions in the Teamcenter workflow before
the actual transfer handler gets executed. It is not used in this demo scenario.

### 4.2.5 t4ea_custom_services_demoerp.sd

This mapping file contains the transfer procedure for the Demo ERP system. The mapping uses the
recommended “Wsdl to T4x” approach.

The call to **::T4X::SOAP::CLIENT::deployShmemVariable** is the part of the result of the T4x
wsdl2t4x utility with the Demo ERP WSDL as input. This call must be placed on top level of any sd file so
that T4x knows the content of the WSDL.

**DemoERPService**: At first the xml payload is prepared according to the parameter operation. If
parameter **::WriteTransferFile2ERPTransferArea** has the value 1 this xml payload information
is written to a file located in the transfer directory **::ERPTransferArea** (both parameters are defined
in file t4ea_mapping_config.sd). Then the web service is called with the T4EA TCL
function **::T4X::SOAP::CLIENT::callSoapOperation** (see the T4EA API Reference for details). At the end the return value of this function is evaluated.

**checkConnection**: This procedure gets called from the custom connection handling whenever a
connectivity check is required (e.g. from a GUI logon dialog or before any transaction starts). Its purpose
is to check if the connection can be established and if the credentials are valid. Since the DemoERP does
not require credentials, the checkConnection procedure only calls a simple echo operation and
returns LOGON_OK on success.

**createTag, createText4Tag and getTagValue**: These are utility procedures for handling XML via the
DOM approach.
4. Mapping Files

4.2.6 t4ea_custom_services_ofbiz.sd

This mapping file contains the transfer procedure for the OFBiz system. Because of the RPC/encoded style of the OFBiz web services, this mapping uses the deprecated “TCL to SOAP” approach.

Note:
It is a special requirement of the OFBiz system that the payload (in soap:body) must contain the credentials. In more common authentication methods the credentials are either passed via HTTP parameters outside of the XML payload or within a XML part specified by e.g. the SOAP standard (soap:header).

ofBizService: First retrieve the credentials for the active connection. Then branch into different procedures depending on the operation.

cHECKConnection: This procedure gets called from the custom connection handling whenever a connectivity check is required (e.g. from a GUI logon dialog or before any transaction starts). Its purpose is to check if the connection can be established and if the credentials are valid. OFBiz provides an echoService operation, which is used to check the connection.

ofBizService4Product, ofBizService4BillOfMaterial, ofBizService4getAssociatedProducts, and ofBizService4Document: These procedures implement the OFBiz specific creation of the XML payload. Then the web service is called with the T4EA TCL procedure ::T4X::SOA::performGenericWebServiceCall (see the T4EA API Reference for details). At the end the return value of this function is evaluated with the common procedure prepareStatus.

prepareStatus: OFBiz-specific handling of the response.

4.2.7 t4ea_custom_functions_oracle.sd

This mapping file contains the transfer procedures for the Oracle database. It uses the T4EA JDBC adapter to access the database. The adapter works with some Java objects that the mapping manipulates via TCL calls. See the namespace ::T4X::OBJECTS in the T4EA API Reference for an explanation of the details. This file is different from the other custom services files since it only implements “high level” procedures. The “low level” connectivity handling happens in t4ea_oracle_connection_handling.sd.

logException: is an auxiliary function to log exceptions from the Java JDBC pipe properly to the session log.

getObjectInfo: is intended to select data from the Oracle table “product_information” for the given ObjectId. With call of ::T4EA::JDBC::ORACLE::getConnectionHandle4System (defined in mapping file t4ea_oracle_connection_handling.sd) a connection to the database is established. After that the statement is prepared and executed. The resulting values are stored in a TCL dictionary named ERPOutputDat.
createProductInformation: With this procedure a data row is inserted into the Oracle table “product_information”. The data information is passed to the procedure by the dictionary inputDict. After connection to the database the next valid product_id is determined. For the demo scenario it is sufficient to seek for the highest used product_id and to increment it. In a productive environment with parallel access to tables it is advised to work with sequence numbers instead. Finally the statement is prepared and executed.

updateProductInformation: With this procedure a specific data row of the Oracle table “product_information” is updated. The needed data information is passed to the procedure by the dictionary inputDict. After connection to the database the statement is prepared and executed.

### 4.2.8 t4ea_oracle_connection_handling.sd

This mapping file implements all procedures needed for the Oracle communication. It implements a kind of connection pooling so that the mapping re-uses the same connection if it tries to connect to the same database with the same user instead of opening a new connection. This sample implementation uses the T4EA managed connection handling to retrieve the connection parameters.

initJdbcConnection: initializes a JDBC Connection.

cleanupJdbcConnection: With this procedure some cleanup activities are done for the current connection like closing JDBC connection, releasing memory allocation.

cleanupAllJdbcConnections: does the cleanup activities for all open JDBC connections.

getConnectionHandle4System: The procedure checks if a JDBC connection already exists and can be reused. Otherwise a new connection is opened. This is the only procedure that should be used from the custom mapping as it opens or reuses a connection automatically as needed.

getConnectionHandle4ActiveSystem: returns the connection handle for the current connection.

createJdbcMetafile: creates the stored procedure metafile for JDBC. This is only relevant for T4O and is unused in the demo scenario.

checkConnection: This procedure gets called from the custom connection handling whenever a connectivity check is required (e.g. from a GUI logon dialog or before any transaction starts). Its purpose is to check if the connection can be established and if the credentials are valid. The demo implementation calls initJdbcConnection to check connectivity and credentials.

::tp_crashexit: internal used procedure which closes all active JDBC connections e.g. in case of a shutdown of the Gateway Server.

### 4.2.9 t4ea_item_import_ofbiz.sd

This mapping file is used to import data from OFBiz into Teamcenter. It is usually triggered by a web service call or script.
The OFBiz import uses the OOTB import type `T4EA_IMPORT_OBJECT`.

**getObjectInfo**: is intended to retrieve data from the external system (via web services) 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. The custom procedure `::T4EA::CUSTOM::SERVICES::ofBIZservice` is used to call the retrieval web service.

**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 `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.

In the demo scenario, the mapping uses the specific query `T4EA_Product_Row` that returns the matching item for a given external ID, if such an item exists.

**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 differ). 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.

**getXMLData**: convenience function that return the value for the given key, if it exists and writes a warning to the logfile and returns an empty string otherwise. This is 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**: unused at the moment.
This mapping file is used to import data from Oracles OE schema into Teamcenter. It is usually triggered by a web service call or script.

Whereas the OFBiz import uses the OOTB import type T4EA_IMPORT_OBJECT, the Oracle import has to define its 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 T4EA_IMPORT_PRODUCT_INFORMATION at the beginning of the file.

**getObjectInfo**: is intended to retrieve data from the external system (via 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. The custom procedure ::T4EA::CUSTOM::ORACLE::getObjectInfo is used to retrieve the record.

**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 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.

In the demo scenario, the mapping uses the specific query T4EA_ProductInformation_Row that returns the matching item for a given external ID, if such an item exists.

**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 differ). 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**: convenience function that return the value for the given key, if it exists and writes a warning to the logfile and returns an empty string otherwise. This is 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**: unused at the moment.

### 4.2.11 t4ea_bom_item_import_template.sd

This mapping file is used to import/update a BOM from an Enterprise Application into Teamcenter. In the demo scenario the import is triggered by the script `var/test/Mapping/t4ea_bom_import_demo.tcl`.

The OOTB import type T4EA_IMPORT_BOM is defined by the call `::T4X::BATCHJOB::IMPORT::CreateImportCodeProxy` at the beginning of the file `var/mmap/t4ea_mapping_config/t4ea_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. In the demo scenario `AdditionalInfo` is equivalent to the file you have prepared for calling the script `t4ea_bom_import_demo.tcl`. In other scenarios without the `AdditionalInfo` it might be necessary to retrieve information from the Enterprise Application within this procedure. 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 T4EA 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 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. 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.

`dummy`: Currently unused.

### 4.2.12 t4ea_query_mapping_demo.sd

This mapping file provides all procedures needed for the external search functionality integrated into Teamcenter.

`executeQuery`: is called by the framework. Depending on the parameter `QueryName` the following procedure is called:

- `::T4EA::QUERY::CUSTOM::MAPPING::executeQuery4OfbizProduct (OFBiz)` or
- `::T4EA::QUERY::CUSTOM::MAPPING::executeQuery4ProductInformation (Oracle database)`

`executeQuery4OfbizProduct`: searches for OFBiz products matching the search criteria. At first the search criteria are stored into dictionary `ERPInputDat`. Then procedure `::T4EA::CUSTOM::SERVICES::ofBIZservice` is called for operation type "performFindList". At the end the search result is written to the framework array `::StatusInfo` in the required format.

`importQueryRowViaBatch`: is called by the framework, if the user presses the "Import Object to Teamcenter" entry in the context menu of the search result. Depending on the parameter `QueryName`
job for import type `T4EA_IMPORT_PRODUCT_INFORMATION` (import from Oracle database) or `T4EA_IMPORT_OBJECT` (import from OFBiz) is created. In the implementation you can switch between direct (synchronous) execution (better feedback to the Teamcenter user) and job (asynchronous) execution (the user’s Teamcenter session is responsive immediately even if multiple objects were selected). The demo mapping assigns a special session ID to the import job, so the job uses the same EA connection that was active during the search. You can use the T4EA connection dialog to change the active connection.

`buildRowQueryDataString`: is used to prepare the required format of the first column of each search result. Teamcenter executes the query specified in this column to determine whether this row maps to a Teamcenter object. If so, the object can be copied and pasted in any Teamcenter application.

4.2.13  `t4ea_prop_mapping_demo.sd`

This file contains all procedures needed for the EA Data view functionality integrated into Teamcenter.

`getXmlData`: is an auxiliary function for procedure `displayOfbizProduct` to get the value for the given key from the ERPOutputDat dictionary, avoiding error messages.

`getArrayDataSave`: is an auxiliary function for procedure `displayOracleProductInformation` to get the value for the given key from the ERPOutputDat dictionary, avoiding error messages.

`displayOfbizProduct`: creates the XML payload for the T4EA data view for an OFBiz product. It calls `::T4EA::CUSTOM::SERVICES::ofBIZservice` with operation type “getProduct” to get the data from OFBiz. Then the following generic functions are used to transform the OFBiz data to the correct format:

• `::T4X::PROP::MAPPING::createMainEntry` (starts the data definition)

• `::T4X::PROP::MAPPING::createSectionEntry` (starts a section)

• `::T4X::PROP::MAPPING::createFieldEntry` (adds a single field with label and value)

• `::T4X::PROP::MAPPING::createSectionEndEntry` (ends a section)

• `::T4X::PROP::MAPPING::createMainEndEntry` (end the data definition)

The procedure `displayOfbizProduct` is not called from the framework directly but only from the demo mapping procedures. It’s sole purpose is to unify the handling of external search objects and imported “real” Teamcenter objects.

`displayOracleProductInformation`: creates the XML payload for the T4EA data view for an Oracle product. It calls `::T4EA::CUSTOM::ORACLE::getObjectInfo` to get the data from Oracle database. This procedure is not called from the framework directly but only from the demo mapping procedures. It’s sole purpose is to unify the handling of external search objects and imported “real” Teamcenter objects.
displaySoapArticle: dummy implementation, not used currently.

displaySoapDocument: dummy implementation, not used currently.

GetEAData_Product: is called by the framework if TargetTypeName “Product” is selected in Teamcenter RAC view “EA Data”. It calls procedure displayOfbizProduct to build the XML payload.

GetEAData_ProductDocument: This procedure is called for the TargetTypeName “ProductDocument”. However there is no implementation to display a file attachment and so this procedure forwards the call to GetEAData_Product and the Product data gets displayed instead.

GetEAData_FixedAsset: is called by the framework if TargetTypeName “FixedAsset” is selected in Teamcenter RAC view “EA Data”. It calls procedure displayOfbizFixedAsset to build the XML payload.

GetEAData_ProductAssociation: is called by the framework if TargetTypeName “ProductAssociation” is selected in Teamcenter RAC view “EA Data”. It reuses the procedure ::T4EA::GENBOM::CUSTOM::MAPPING::getERPBOMLine from the ProductAssociation BoM Mapping to retrieve the BoM structure and procedure ::T4EA::CUSTOM::SERVICES::ofBIZservice to retrieve data for each BoM position and then builds the XML payload for the data view including a table.

GetEAData_ProductInformation: is called by the framework if TargetTypeName “ProductInformation” is selected in Teamcenter RAC view “EA Data”. It calls procedure displayOracleProductInformation to build the XML payload.

GetEAData_Article: would be called by the framework if TargetTypeName “Article” could be selected in Teamcenter RAC view “EA Data”. In the demo scenario “Article” is not configured for selection.

GetEAData_Document: would be called by the framework if TargetTypeName “Document” could be selected in Teamcenter RAC view “EA Data”. In the demo scenario “Document” is not configured for selection.

GetEAData_BillOfMaterial: is called by the framework if TargetTypeName “BillOfMaterial” is selected in Teamcenter RAC view “EA Data”. It calls procedure ::T4X::SOA::performGenericWebServiceCall to get the BOM view definition and creates the XML payload for the T4EA data view.

GetEAData_QueryResultObject: creates the XML payload for the T4EA data view for an external query object (a row in the search result of an external search). It calls procedure displayOfbizProduct in case of QueryName “T4EA_Product” and procedure displayOracleProductInformation for QueryName “T4EA_ProductInformation”.
4.2.14  t4ea_idgen_mapping_template.sd

This file contains the configuration for retrieving/verifying a Teamcenter number (ID) from an external system.

getReservedFixedAssetNumber: The procedure calls ::T4EA::CUSTOM::SERVICES::ofBIZservice with operation type "createFixedAsset" to get the next free object number from OFBiz.

checkIfFixedAssetExists: The procedure calls ::T4EA::CUSTOM::SERVICES::ofBIZservice with operation type "performFindList" to check if the given ObjectId already exists in OFBiz.

getReservedNumber: is called by the framework. The procedure returns for the given parameter ObjectType the corresponding next free object number generated by the external Enterprise Application. In the demo scenario the procedure ::T4EA::IDGEN::CUSTOM::getReservedFixedAssetNumber is called for ObjectType "EAX4FixedAsset".

checkIfObjectExists: is called by the framework. The procedure checks if for the given parameters (ObjectType and ObjectId) an item already exists in the external Enterprise Application. In the demo scenario the procedure ::T4EA::IDGEN::CUSTOM::checkIfFixedAssetExists is called for ObjectType "EAX4FixedAsset".

4.2.15  t4ea_connection_mapping_template.sd

This file contains the custom implementation of the T4EA managed connection handling for your EA systems. It maps EA connections to TargetTypeNames, Query names and import types and vice versa by the TCL dictionary EASystemMap. It also shows a possibility to handle unmanaged connections in a managed environment (TCL list IgnoredTargetTypeNames).

checkConnection2EA4Session: Given a session ID and the TargetTypeName, this procedure must determine if the active connection is valid for the intended action and if so make sure the connection is established.

checkConnection2EA4Transaction: Similar to checkConnection2EA4Session this procedure must check if the active connection is valid for the intended transaction.

isIgnoredTargetTypeName: helper procedure (not called from the framework but only from other custom procedures) to determine whether the given TargetTypeName needs to be handled by the T4EA EA managed connection handling (returns FALSE) or is ignored by the T4EA EA connection handling ("plain connection handling", returns TRUE).

checkActiveEASystemForTargetTypeName: given a TargetTypeName this procedure checks if the active EA connection is valid.
getTargetTypeNames4Connection: given an EA connection identifier, this procedure must return all TargetTypeNames, search names and import types that this connection can handle. This procedure is used by the GUI logon dialog to display only relevant connections.

connectEA: this is the central handler for login attempts. Given the connection parameters and the credentials, it must hand them over to the specific connection procedures (checkConnection in several namespaces in the demo configuration) depending on the given EA connection identifier. In the demo this procedure calls the connect procedures as configured in the EASystemMap dictionary to achieve this.

4.2.16  t4ea_async_services.sd

This file is not sourced by default as it contains a demo for an advanced asynchronous communication scenario using SOAP with a callback service. T4EA calls an “external” service which is implemented by T4EA in the demo. The demo can be started by the procedure startDemoAsyncCall.

This service (named :T4EA::ASYNC::SERVICES::dummyAsyncService) takes some payload information (an item ID) and some information necessary to set up the asynchronous communication (the callback URL and a unique message ID). It then sets up a T4EA job, just to simulate some asynchronous control flow.

The last action of the created job calls the callback URL and passes some processing result. This will cause the original caller (startDemoAsyncCall) to end the wait state and proceed with its work. So the asynchronous communication is wrapped into a synchronous control flow. startDemoAsyncCall also implements some timeout feature which can easily be adapted.

Note that the callback service and the procedure communicate via a so-called proc flag which allows different active instances within a T4EA Gateway Server and even different T4EA Gateway Servers (connected to the same Basic Gateway Server) to communicate and to synchronize.

After sourcing the mapping file you can execute the scenario using the script “T4EA demo asynch service test” in the “Examples” group of the T4EA script environment in the T4EA Gateway Service Admin GUI. Enter some ID and the URL the service should use for the callback, e.g. http://t4eags:11301/soap/callback.

4.2.17  t4ea_meta_import_template.sd

This file is not sourced by default as it contains a namespace template for the meta import batch jobs, used to combine other import jobs into one bigger job.

4.2.18  t4x_mapping_config.sd

This file just sources the other three t4x mapping files in order to build the rfdt package.
4.2.19 t4x_custom_mapping_toolbox.sd

This file contains the two central procedures selectTC_Object and createTC_Object which are called in other mapping files, e.g. t4ea_item_import_ofbiz.sd, to search for Teamcenter objects and to create new Teamcenter objects. The procedures use the new Teamcenter ITK functions to support the “Multi Field Key” feature of Teamcenter.

4.2.20 t4x_user_exit_template.sd

This file contains the two procedures beforeTransfer and afterTransfer which can be used to perform additional actions (user exit calls) before and after each major/data changing ERP call (only relevant for T4S and T4O) as well as before and after each JCO call (only relevant for T4S). The procedures are not used in the T4EA demo scenario.

4.2.21 t4x_workflow_batchjob_template.sd

This file contains a procedure to send a Teamcenter e-mail. The procedure can be called by workflow jobs.
5. How to use the demo scenario

5.1 Connection Handling

For some demo use cases you have to take care to select the correct connection. To open the T4EA EA connection dialog, choose “EA Connections” from the “T4EA Gateway” menu:

This will open the connection dialog with four preconfigured connections:

The connections “OFBiz1”, “DemoERP1” and “Oracle1” are configured for auto login in t4ea_mapping_config.sd. That is why they appear with the green symbol in the status column. The “Oracle2” connection is configured without credentials in t4ea_mapping_config.sd, therefore the red icon appears in the status column. Before using this connection, you must first login. T4EA will open the login dialog automatically if required. You can explicitly login by clicking the EA logon symbol in the “Action” column. The checkbox in the “Preferred” column is set to “Oracle1” by default. You can change the preferred connection by simply clicking on the desired connection.

5.2 Transfer to OFBiz

Create a standard item or open an existing item in My Teamcenter and select the item revision.
Press “Ctrl-P” or select “File/New/Workflow Process”. Select the “Process Template” “T4EA_Product_Transfer”.

Click “OK”. The workflow will start and the item revision will be transferred to OFBiz. The “New Process” window should close without any error message. You do not have to change the active connection explicitly, because T4EA knows that the “OFBiz1” connection is to be used by the workflow configuration.
Click the item revision master form and open the “Viewer” tab:

Note that the “Item Comment” field contains the date and status of the last transfer. In case of success, the “User Data 1” field contains the ID of the Product in OFBiz.

Now review the transaction log by opening your T4EA Basic Gateway Service Admin GUI. After logging in, select “Log files/Transaction” in the menu.
Enter some search criteria in the “search” field, e.g. “Bearing” and click the magnifier.

Click on the green entry on the left.

Review the log file and the transferred values. You can click the “Session file” link to see details on the transfer.
Verify the result using OFBiz’ catalog manager. Log in to OFBiz (user “admin”, password “ofbiz”) https://ofbizserver:8443/myportal/control/checkLogin and confirm the security warning by adding the OFBiz URL to the list of trusted SSL sites.

Enter some keyword, e.g. “Bearing” in “Search Products/Keywords” and click “Find”.

![Image of OFBiz catalog manager interface]
Click on one of the search results to see details on the product and verify the values.
5.3 Transfer to Oracle

If you want to transfer item revisions to the Oracle OE schema instead of OFBiz, follow the first steps of Transfer to OFBiz, but choose the workflow named “T4EA_ProductInformation_Transfer”. The transfer will store the ID of the created table row in the field “User Data 3” and of course you have to query the product_information table to review the result. You can use the same Teamcenter object and transfer it either to OFBiz, Oracle, DemoERP or all of them.

The demo scenario has two Oracle connections “Oracle1” and “Oracle2” that are both valid for the ProductInformation transfer. The workflow is configured to display the valid connections and ask the user which one to use (look at the OFBiz transfer to see how to use a dedicated connection). Once you start the workflow, it will either complete if Oracle1 or Oracle2 was selected active, or otherwise it will wait with a message:

```
Warning

The initial action was successful, but a subsequent task in the workflow may have failed. For additional details, please read the rest of the error messages.

The initial action was successful, but a subsequent task in the workflow may have failed. For additional details, please read the rest of the error messages.
Business rules for handler 'T4EA-validate-EALogon' on action 'Complete' in task 'Logon to Oracle' are not met.
Active EA connection does not match intended action.
```

In this case a task will appear in your worklist and if you change to “My worklist” and select the logon task, you can choose which Oracle connection should be used and enter credentials.

The transfer will then use the credentials entered by the user instead of the preconfigured ones. To verify the transfer you can review the table entries in the database with an SQL query, e.g.:
5.4 Transfer to Demo ERP

For transferring data to the DemoERP, start the workflow “T4EA_Article_Transfer”. The workflow is configured in a way that selects the target ERP system (“DemoERP1”) directly, so there is no need to select the EA connection. DemoERP1 is also configured for auto login, so no login task will appear and the workflow will complete immediately. Since this connection only communicates with a dummy implementation, there is no real verification of the transfer besides the review of the log files and the validation of the transferred XML in SoapUI.

Note:

T4EA_Article_Transfer is an example of a workflow that restores the user’s preferred connection after finishing the workflow while setting a specific connection during the workflow transfer. This can be done using the T4EA-SessionProxy handler with argument “-mode=SAVE” to save the preferred connection and to restore it with argument “-mode=FORGET” in case of success and errors.

5.5 Query OFBiz and Import

Select “OFBiz1” in the connection dialog:

In My Teamcenter, open the “Search” view by clicking the magnifier in the tool bar.
Click on the “Select A Search” button in the Search view and click “More...“.

Select the “T4EA_Product” search and click “OK”. Enter some search criteria (e.g. “*test*”) and click the (green) button to execute the search. The “Search Results” view will open.
Select one of the search results. Review the ERP data in the “Viewer” tab. Right click on one of the search results opens the context menu: select “Import Objects to Teamcenter”. The import will run in the background, using the T4EA job queue. Wait about 30 seconds.

You can watch the progress of the job using the T4EA Basic Gateway Service Admin GUI. Click “Job Management/Jobs”, type “T4EA” in the search criteria field and press the magnifier.
The last image shows a job that has completed successfully. You can review the transaction log by clicking on “Log file” or change job parameters by clicking the pencil icon.

Back in the Teamcenter RAC, select the result, for which you just initiated the import and click Ctrl-c. This will copy the Teamcenter representation of the ERP object, which was just created by your import job, to the Teamcenter clipboard. Verify the clipboard content by clicking the clipboard icon in the lower right of your Teamcenter RAC.
If you see a blue ball (the representation for an external, non-Teamcenter object), the import has not yet finished or was unsuccessful. If the clipboard contains an internal object (in this case an item or a revision, depending on your mapping), you can paste that e.g. to your Teamcenter Home folder and continue working with that internal object.

5.6 Query Oracle and Import

This is similar to OFbiz, with the only difference that you must select either “Oracle1” or “Oracle2” in the T4EA EA connection dialog and must execute the “T4EA_ProductInformation” query in Teamcenter.

5.7 EA Data View

For any item previously transferred to or imported from any demo Enterprise Application or for search results (blue balls) from a T4EA external search, you can display data from the external application within the Teamcenter RAC.

In the Teamcenter RAC with the demo scenario, just click “Show EA data” in the “T4EA Gateway” menu of “My Teamcenter”. Generally, open menu “Window”, entry “Show View/Other”. In the “Teamcenter Gateway” group, select view “EA Data” and click “OK”.

![Clipboard Contents](image)
This will open the EA data view. In My Teamcenter, select a Teamcenter item revision associated with some external system and review the displayed data.
You can switch between different TargetTypeNames by choosing from the left drop-down box: In the demo scenario “Product” stand for the OFBiz system, “ProductInformation” for the Oracle OE schema. The right drop-down box lets you choose from applicable connections, in case of the demo scenario “Oracle1” and “Oracle2” are available.

### 5.8 Import script for BOMs

The demo script for importing BOMs contains a sample file which contains a sample for a flat file which can be read by the script and passed to the import mapping. This script can be used on its own or used in combination with the T4x GS scheduler to automatically read BOM import files whenever they appear at a certain directory.

The file format looks like the following sample:

```plaintext
# First line contains header item ID:
10000
# PositionNo, PositionType (unused here!), MaterialNo (ItemID), Quantity, CadIndicator
5           FERT                         1000010              10        X
10          FERT                         1000013               3        X
15          FERT                         1000012               2        X
```

The file contains some comments (indicated by the # in the first position), the item ID of the BOM header in the first non-empty, non-comment line, and three single-level BOM lines which should be imported as children of the header item. The columns of the positions are space-separated. The IDs in the “MaterialNumber” column are the IDs of Teamcenter items. The items must not exist prior to the import. The values in the “PositionNo” and “Quantity” columns will set the corresponding OOTB
Teamcenter occurrence notes for the BOM lines. Note that the file reader ignores empty lines and lines with a leading “#” (comments).

To start an import via script, you must first create a file according to the given format.

Open the T4EA Gateway Service Admin GUI and select “Script/Scripts”. Select “T4EA BOM Import Demo”.

Type in the file name of your input file in “INPUT FILE”. Mode “Direct” will execute the import directly and immediately, mode “Send to Job Pool” will create a job on the BGS instead and the job will be executed by a T4x GS job client, if configured. For now, choose “Direct”. Click “Run script”.

---

Teamcenter Gateway for Enterprise Applications Demo Guide
© 2018 Siemens Product Lifecycle Management Software, Inc.
Note that the demo BOM import does not use a managed EA connection. So it is not required to select a certain EA connection before starting the script.

The line

```
XXXXXXXXXXXXX ImportStatus = OK
```

indicates that the import was successful. You can switch to the “Log file” tab and review the results in detail.

In Teamcenter RAC, select the header item revision and press F5 (refresh) to see the updated object. You will notice that the item revision now contains a BOM view revision.
Select the revision of the head item and right click it. Choose “Send to...” from the context menu and select “Structure Manager”.

Verify that the occurrence notes “FindNo.” and “Quantity” are set appropriately.

You can now change the contents of the input file (add or drop lines, change occurrence note values) and restart the script (with the same head item ID or different IDs) and verify the results in Teamcenter.
To automatically import incoming files, configure the T4x GS scheduler to execute the script repeatedly after a configurable period of time. Open the GS Admin GUI and open “Configuration/Task Management”, then click the “plus” icon:

Choose “tpapps.exe”, enter var/init/start.tpscript var/test/Mapping/t4ea_bom_import_demo.tcl -File C:/TransferArea/t4eaimport in the “Parameter” field (replace C:/TransferArea/t4eaimport with the name of a directory on your host). Click “Run Process regularly” and choose a period, e.g., “Every minute”. Press “Apply” and then “Save whole configuration” (check on the upper right). The script assumes that the directory contains the sub-directories “new”, “inwork”, “error” and “processed”. New files should arrive in “new” and are moved accordingly by the import script.

The scheduler will begin to repeatedly execute the script. Whenever one or several files are in the “new” directory, it will move the file to “inwork” and create a job for each file. Afterwards it will move the file to “processed” or “error”, according to the success of the job creation. You can watch the activity of the scheduler in the system log file tpscript.log.
5.9 Use OFBiz to generate Teamcenter Item ID

Create an item of type "FixedAsset" in Teamcenter. In the creation dialog click on the "Assign" button. At this moment T4EA calls OFBiz to generate the next free number for a "FixedAsset". OFBiz creates a "FixedAsset" dummy object of type "OTHER_FIXED_ASSET" with name "Reserved by Teamcenter". The id is sent back to Teamcenter and displayed in the ID field. Specify name and description and click "Finish".

Open the EA Dataview and select the item revision of the new created object. Then select the TargetTypeName "FixedAsset" (left drop-down box) in EA Dataview. The data of the corresponding dummy OFBiz object are displayed.
Press "Ctrl-P" or select "File/New/Workflow Process". Select the process template "T4EA_FixedAsset_Transfer" and click "OK". The workflow will start and the item revision will be transferred to OFBiz. In OFBiz the corresponding object is renamed to the Teamcenter ItemRevision name ("MyFixedAsset" in the sample screenshot).

For more information how to configure ID generation see Teamcenter Gateway - Generic Configuration Guide.
5.10 Verify Teamcenter Item ID in OFBiz

For this demo scenario the value of Teamcenter preference "T4X_IDGEN_Function4EAX4FixedAsset" has to be set to "T4X_IDGEN_EXTERNAL_VERIFY_OF_NUMBER".

Create an item of type "FixedAsset". In the creation dialog click on "Assign" button. In this case an item ID (pattern "AAnnnnnnnnn") is generated by Teamcenter and verified in OFBiz searching for a FixedAsset with same ID.

```
Specify name and description and click "Finish". Select the item revision of the new created object and press "Ctrl-P" or select "File/New/Workflow Process". Select the "Process Template" "T4EA_FixedAsset_Transfer" and click "OK". The workflow will start and the item revision will be transferred to OFBiz using the ID created by Teamcenter.
```
For more information how to configure ID verification see Teamcenter Gateway - Generic Configuration Guide.
A. Glossary

A

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.

Apps
See "GS".

B

BGS
Basic Gateway Service.

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.

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.

D

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 Data View mark-up.

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.

**G**

**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 Teamcenter and the Enterprise Application.

**GUI**
Graphical user interface.

**I**

**IDGEN**
The IDGEN is a mechanism to get an external ID from the ERP system when assigning a Teamcenter ID.

**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**

**JDBC**
Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database.

**L**

**LOV**
List of Values.
M

MFK
Multi-key functionality in Teamcenter.

O

OOTB
Out of the box.

R

RAC
stands for Rich Application Client also referred to as rich client or portal.

S

SSL
Secure Sockets Layer.

T

T4x
The entire Teamcenter Gateway product family.

TC
Teamcenter

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

TEM
Teamcenter Environment Manager.

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. URLs are
often used to identify configurations in Java and other languages. See [https://en.wikipedia.org/wiki/Uniform_Resource_Identifier](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 of URNs.

**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**

**Z-Table**
“Z” is the prefix name for custom tables well-known in SAP world.
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.

© 2018 Siemens Product Lifecycle Management Software Inc. Siemens, the Siemens logo and SIMATIC IT are registered trademarks of Siemens AG. Camstar, D-Cubed, Femap, Fibersim, Geolus, I-deas, JT, NX, Omneo, Parasolid, Solid Edge, Syncrofit, Teamcenter and Tecnomatix are trademarks or registered trademarks of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries. All other trademarks, registered trademarks or service marks belong to their respective holders.