|  |  |  |  |
| --- | --- | --- | --- |
| **APPROVAL LIST** | | | |
| **Name** | **Company** | **Function** | **Date** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **MODIFICATIONS** | | | |
| **Issue** | **Date** | **Modified by** | **Observations** |
| x.y | 01/01/2017 |  |  |
|  |  |  |  |
|  |  |  |  |

© **EADS** 2017. All rights reserved. **Confidential and proprietary document**.

© **EADS** 2017. All rights reserved. **Confidential and proprietary document**.

**TABLE OF CONTENT**

[1 Introduction 4](#_Toc488270171)

[2 Purpose 4](#_Toc488270172)

[3 Prerequisites 5](#_Toc488270173)

[3.1 Load [scm] components 5](#_Toc488270174)

[3.2 Load [framework] components 8](#_Toc488270175)

[4 Templates 10](#_Toc488270176)

[4.1 Template [Generic Middleware Connector] 10](#_Toc488270177)

[4.1.1 Application Design 10](#_Toc488270178)

[4.1.2 Application Example 10](#_Toc488270179)

[4.1.3 Template Design 10](#_Toc488270180)

[4.1.4 Template usage 12](#_Toc488270181)

[4.2 Template [SAP Middleware Connector] 21](#_Toc488270182)

[4.2.1 Application Design 21](#_Toc488270183)

[4.2.2 Application Example 22](#_Toc488270184)

[4.2.3 Template Design 22](#_Toc488270185)

[4.2.4 Template usage 25](#_Toc488270186)

[5 Application examples 31](#_Toc488270187)

[5.1 Example – Generic Middleware Connector 31](#_Toc488270188)

[5.1.1 Introduction 31](#_Toc488270189)

[5.1.2 Prerequisites 31](#_Toc488270190)

[5.1.3 Distribution 31](#_Toc488270191)

[5.1.4 Routing setup 32](#_Toc488270192)

[5.1.5 Run the [example-mc-genapp] example 33](#_Toc488270193)

[5.2 Examples – SAP Middleware Connector 35](#_Toc488270194)

[5.2.1 Introduction 35](#_Toc488270195)

[5.2.2 Prerequisites 35](#_Toc488270196)

[5.2.3 Distribution 36](#_Toc488270197)

[5.2.4 Routing setup – SAP logical Connector 40](#_Toc488270198)

[5.2.5 Run the complete SAP Connector Application 40](#_Toc488270199)

# Introduction

This guide contains the description of the detailed steps involved in the development of TIBCO BW6.x applications in the context of the SANOFI TIBCO ESB Factory. The SANOFI TIBCO ESB factory maintains standard practices and tools allowing for an efficient and reliable application development using a standard framework and process templates collection.

The guide has been written in the form of a step-by-step tutorial that will drive a developer through the referenced application designs, examples and services generation from templates.

# Purpose

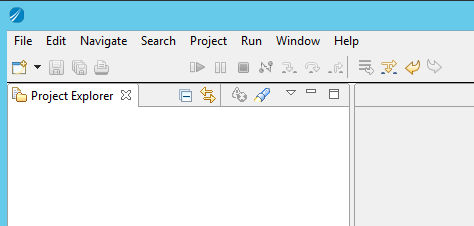
The purpose of this document is to offer SANOFI TIBCO ESB services developer a fast support in enabling essential TIBCO application and services using the standard SANOFI ESB practices and framework.

# Prerequisites

Developers must have access to a standard SANOFI TIBCO ESB Development host (Citrix VMI). The developer workstation is equipped with all necessary TIBCO and 3rd party software. The TIBCO Studio is preconfigured with the SANOFI GIT Repository connectivity.

## Load [scm] components

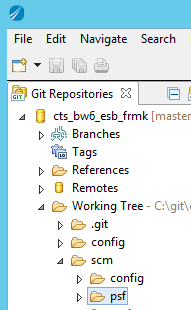
Your TIBCO Studio has been configured with a default Workspace containing no projects.



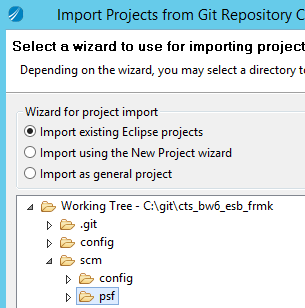
All component sources will be accessed using Eclipse psf (Project Set File). The GIT Repository contains the collection of all PSF files for every existing component.

**You first need to import the scm psf collection into your workspace.**

* Navigate to the GIT Repository and locate the SANOFI GIT entry.
* Open the “Working Tree/scm” folder, and select the psf folder.



* Right click the “psf” folder and select “Import Projects”
* Keeping the “psf” folder selected (bottom tree), check the “Import existing Eclipse project” option (upper check boxes).

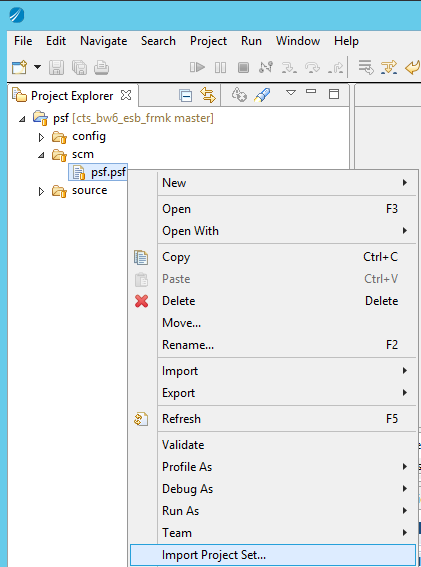


* Click “finish”

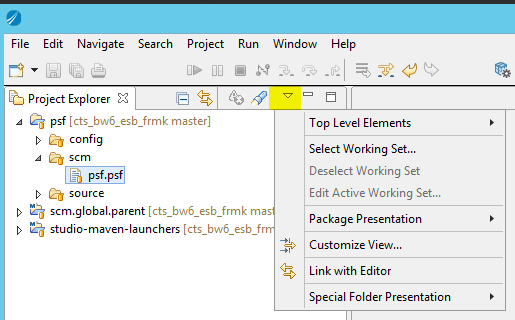
[This creates a new project in your studio workspace;

**You will now use the [psf] distribution to complete the loading of the [scm] assets, including projects and related eclipse working set configuration.**

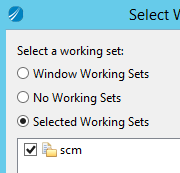
* Select the “psf/scm/psf.psf” file and right click, then select “Import Project Set…”.



* This completes the loading of the essential [scm] assets. Activate the “Working Set” using the view menu:

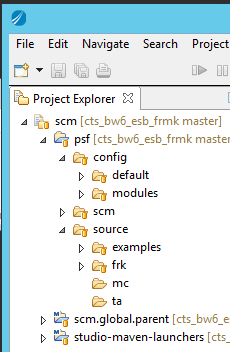


* Select “Top Level Element” / Working Sets.
* Go back to the view menu, and select “Select Working Set”. Then check “scm”.



* Press ok. This will display the [scm] working set with the referenced eclipse projects.

*Note: the “scm/psf” folder contains all the project set files relate to [sources] and [configuration] distribution. This is where a developer goes to find and open a specific [module] source and/or configuration.*



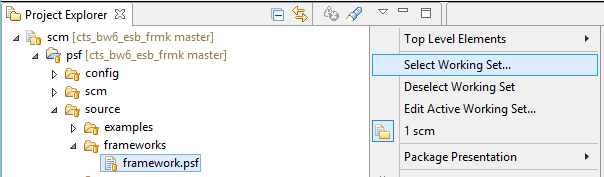
You have loaded the [scm] distribution into the TIBCO Studio.

## Load [framework] components

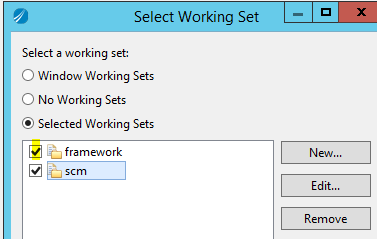
In order to develop BusinessWorks 6.x application, the SANOFI TIBCO BW framework must be pre-loaded in the TIBCO Studio.

**You will now load the framework assets using the preconfigured [psf].**

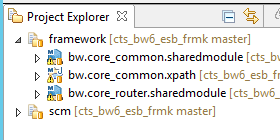
* In your Studio “Project Explorer”, Navigate to the “scm/psf/source/frk/framework.psf” file, right click, and select “Import Project Set”.
* Go to the view menu the “Select Working Set”



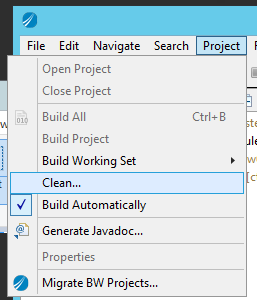
* Then select th “framework” Working Set, and click OK.



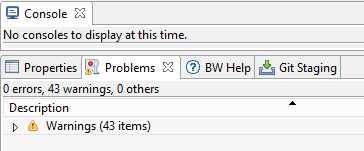
* You now have loaded the framework components in the workspace

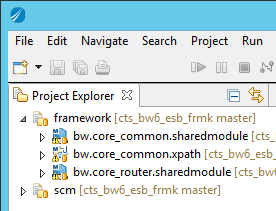


* In studio, run a “Project/clean, clean all projects, OK



* No errors should be displayed in the Studio “Problems” tab;





You have loaded the ESB framework distribution in the Eclipse Studio.

# Templates

## Template [Generic Middleware Connector]

### Application Design

The generic middleware connector templates are used to generate basic MC (Middleware Connector) TIBCO BusinessWorks 6.x application and services according to the predefined SANOFI standard ESB framework design policies. The Connector services can be of different types:

* MCI: Middleware Connector In: asynchronous messaging from an integrated application to the ESB.
* MCO: Middleware Connector Out: asynchronous messaging from the ESB to an integrated application.
* MCS: Middleware Connector Service: Request/reply interface exposed on the ESB implementing request logic on the integrated application.

*Note [July 20th 2017]: MCS interface is not yet in the template*



### Application Example

An example of a generated application is described in section 5.1.

### Template Design

#### Architecture

The template distribution is based on a fully featured connector application where special <keywords> are used for the replacement with variable parameters provided by the user. The parameters are automatically replaced by the template engine (maven) during the application generation. There are three actions currently available in the [Generic Middleware Connector] application template:

|  |  |  |
| --- | --- | --- |
| **Template Generic Middleware Connector** | | |
| **Function** | **type** | **Description** |
| template-gen-application-mc | Template | Generates a new empty generic middleware connector application |
| template-feature-gen-application-mci | Template feature | Generates a new MCI operation inside an existing connector application. |
| template-feature-gen-application-mco | Template feature | Generates a new MCO operation inside an existing connector application. |

The below diagram shows the template architecture and the variable parameters.



#### Module name qualification

The “module name” is the name of the DEVOPS “module” the connector is to be member of. A [module] is just a top level folder containing one or many applications to be developed and released together in the same stream/visioning. You find the module folder in the GIT repository immediately under the “source” folder.

A module name can be defined as:

[ModuleName]: low-case string defining the overall functionality involved in the embedded applications.

If the module is designed to contain ONLY one application, you can give the module name the same name as the intended application name (see next section).

#### Logical Connector name

The logical connector name is based on the following naming standard. The naming allows for defining application connector name based on the integrated application name and the variable scope of the service(s) to be implemented in the connector. The [LogicalConnectorName] will be used in many items naming in the subsequent TIBCO BW6.x project content.

[LogicalConnectorName]: mc-<integrated application name>(-<function scope>)\*

<integrated application name>: name of the application the connector is designed for; example “siebel1”

<function sub-scope>: optional specific -businessDomain, -objectType, -objectName, -operation, defining a specific scope for the services in this connector.

Example:

* mc-siebel1
* mc-siebel1-CR: connector for application Siebel1 - all objects.
* mc-siebel1-CR-prospect: connector for application Siebel1 - prospect services.
* mc-siebel1-CR-prospect-address: connector for application Siebel1 - prospect category, address object.
* mc-siebel1-CR-prospect-address-update: connector for application Siebel1 – prospect category, address object, update operation only.

#### Template properties.

The table below summarizes the properties to input for each template instance. Properties that are not displayed in the table

*Note: currently, the template configuration exposes the low level parameters required for the maven based generation. Later enhancement using eclipse plugin/form will allow more abstraction.*

Please, refer to the [ModuleName], [LogicalConnectorName] defined in the above section 4.1.3.2, 4.1.3.2 to use in the various properties below.

|  |  |  |
| --- | --- | --- |
| **Template Generic Middleware Connector properties and naming standard** | | |
| **Function** | **Template Property** | **Description** |
| template-gen-application-mc | module | [ModuleName] |
|  | artifactId | mc-[LogicalConnectorName] |
|  | genConnectorScope | [LogicalConnectorName] |
|  | appModuleName | mc-[LogicalConnectorName] |
| template-feature-gen-application-mci | module | [ModuleName] |
|  | artifactId | ww.mc-[LogicalConnectorName] |
|  | genConnectorScope | [LogicalConnectorName] |
|  | object-action | Name of the object and optional operation describing the MCI service to add.  Use naming: <object>(-sub-action)\* |
|  | taskCodeMCI | Referenced MCI task code as provided in specification.  Naming: MCI-nnnnn |
| template-feature-gen-application-mco | module | [ModuleName] |
|  | artifactId | ww.mc-[LogicalConnectorName] |
|  | genConnectorScope | [LogicalConnectorName] |
|  | object-action | Name of the object and optional operation describing the MCI service to add.  Use naming: <object>(-sub-action)\* |
|  | taskCodeMCO | Referenced MCO task code as provided in specification.  Naming: MCO-nnnnn |

### Template usage

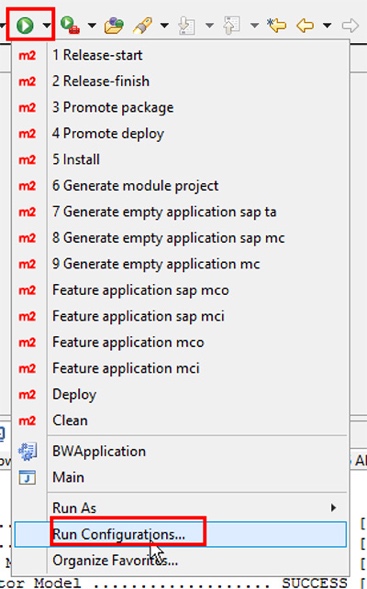
In this tutorial, you will generate a complete Middleware Connector BusinessWorks application.

#### Module generation

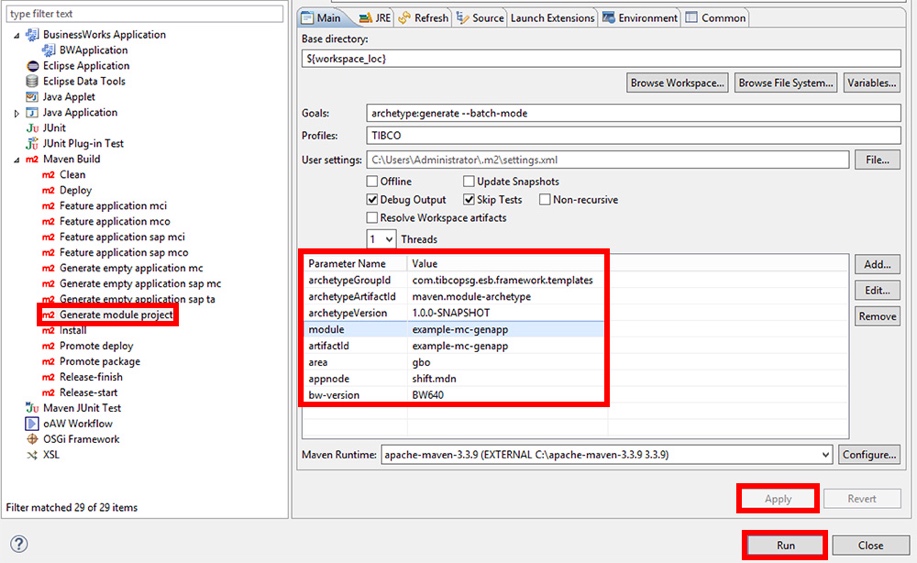
If your [module] does not exist, then generate it.

**You will now generate the module hosting your project.**

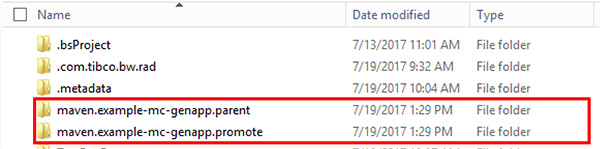
* Select m2e shortcut, and click on “Run Configurations…”:



* Select the “Generate module project” Maven build, and update the values of these fields:
  + module
  + artifactId
  + area
  + appnode
  + bw-version
* Apply
* Click on Run



* Two projects are generated in the Studio workspace:



* We will import these projects later in the Studio

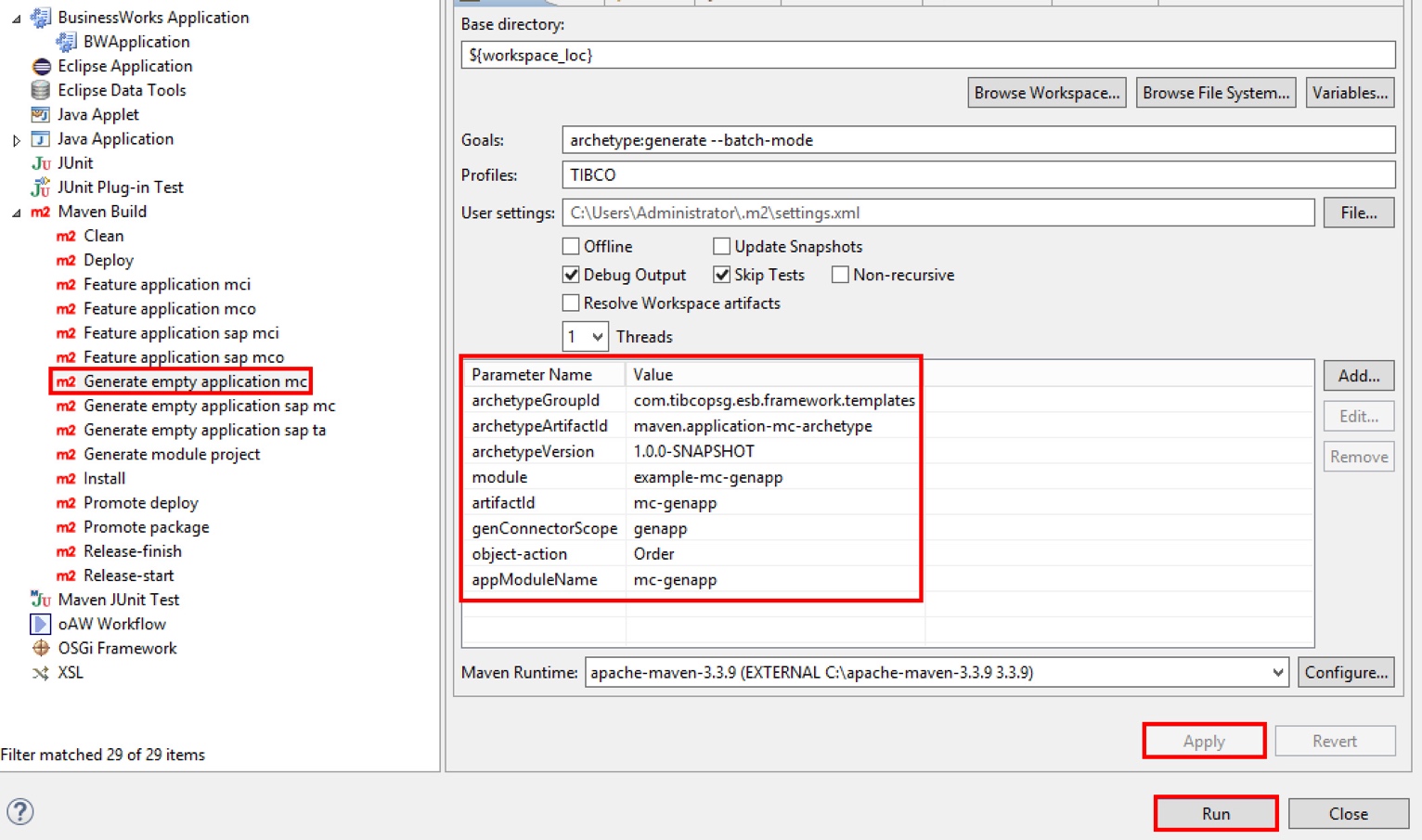
You have generated your application(s) module container.

#### Connector application generation

If your connector application does not exist already, then you can generate an empty connector application by following the steps in this section.

**You will now generate a middleware logical connector (with no service)**

* Select the “Generate empty application mc” Maven build, and update the values of these fields:
  + module
  + artifactId
  + genConnectorScope
  + appModuleName
* Apply
* Click on Run



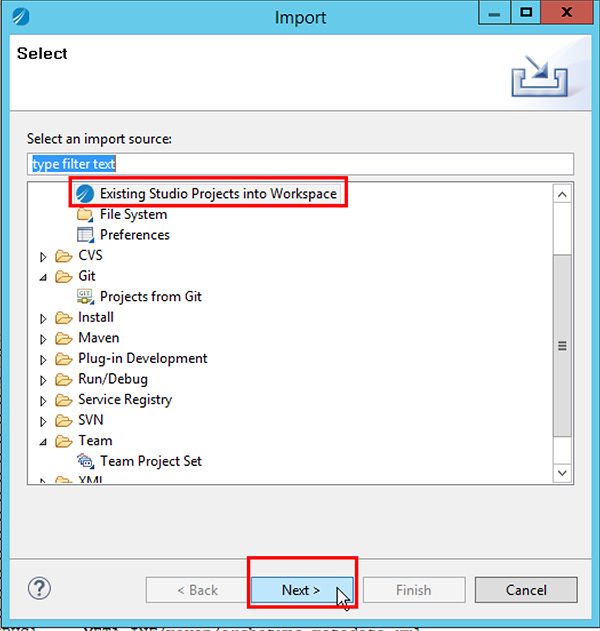
* Three new projects are generated in the Studio workspace

You have generated the logical connector application projects.

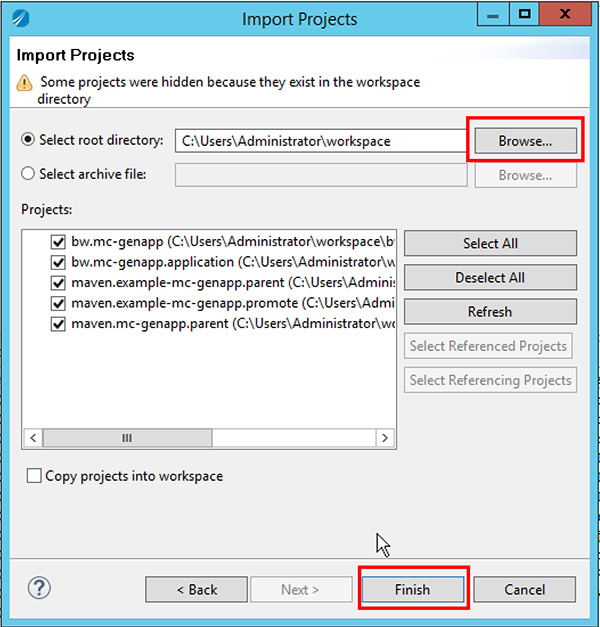
#### Import the Connector application generated projects into your workspace.

**You will now import the generated projects into the Studio workspace.**

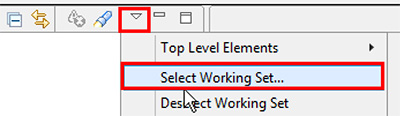
* In Studio select:
  + File > Import
  + Existing Studio Projects into Workspace



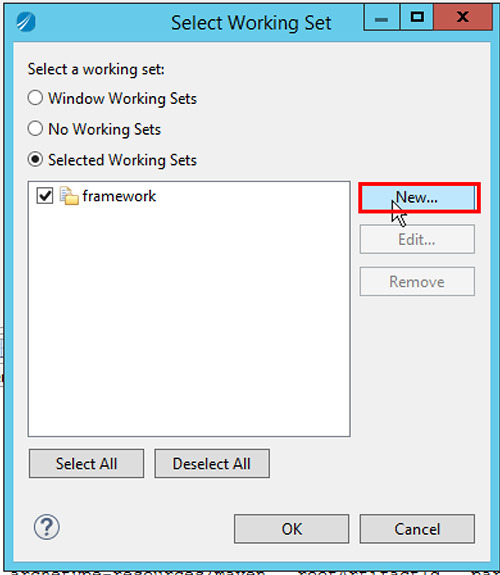
* Browse to Studio workspace location:



* Projects are imported to the Studio workspace, but are not yet visible. You need to add them in a working set.



* Add a new working set



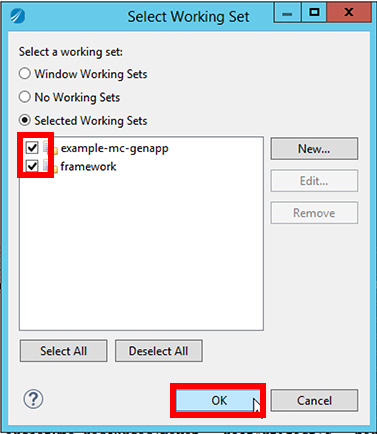
* Select “Resource”



* Set a name, use the module name
* Select the previously generated and imported projects



* Select all working sets





* Open parent project pom.xml file and uncomment the commented lines:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<artifactId>maven.mc-genapp.parent</artifactId>

<packaging>pom</packaging>

<parent>

<groupId>com.sanofi.esb.example-mc-genapp</groupId>

<artifactId>maven.example-mc-genapp.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../maven.example-mc-genapp.parent/pom.xml</relativePath>

</parent>

<modules>

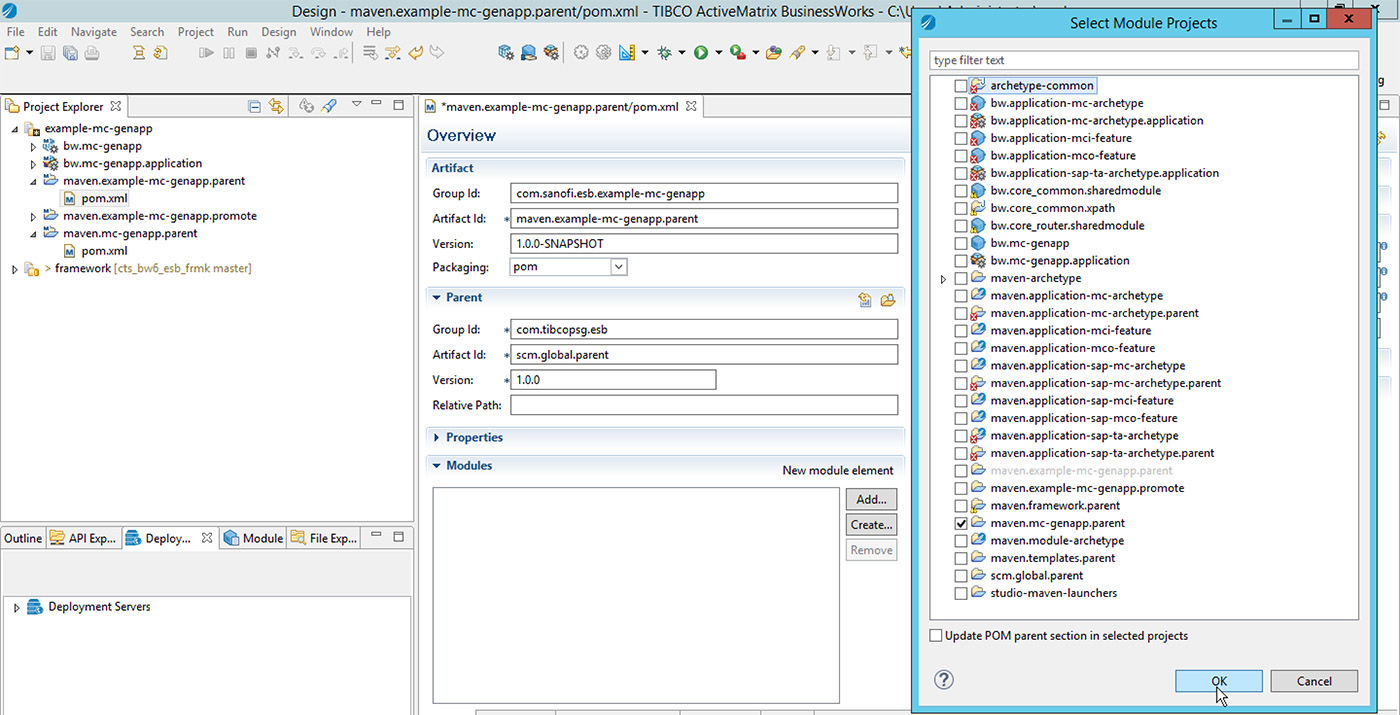
<module>../bw.mc-genapp</module>

<module>../bw.mc-genapp.application</module>

</modules>

</project>

* Finally open the module parent project pom.xml and add the two BusinessWorks projects in modules:



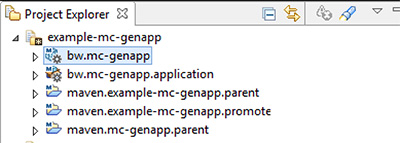
You have imported the generated projects into a Studio working set.

#### Add a Connector MCI service

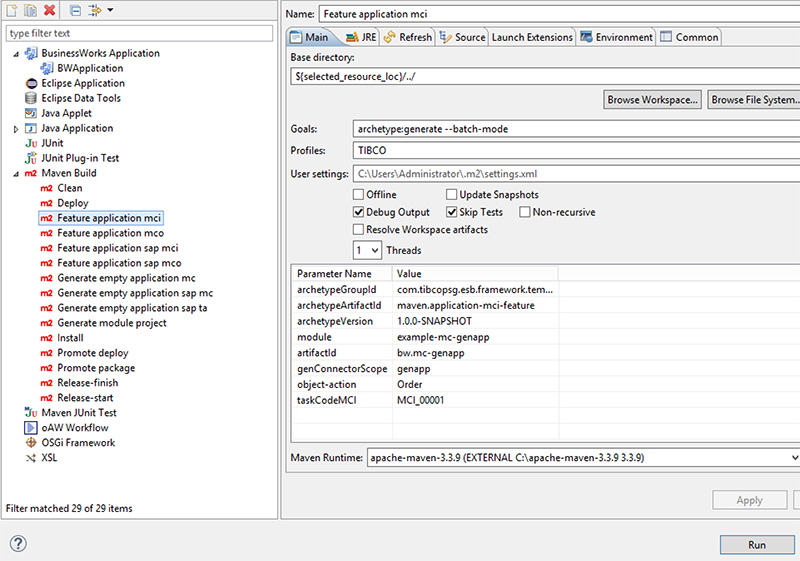
If you need to add an MCI connector service, please follow instructions in this section.

**You will now add logical connector MCI service in an existing Connector application.**

* Select the application module that you previously generated



* In the Maven builds, select the “Feature application mci”, and fill the values of these fields:
  + module
  + artifactId
  + genConnectorScope
  + object-action
  + taskCodeMCI



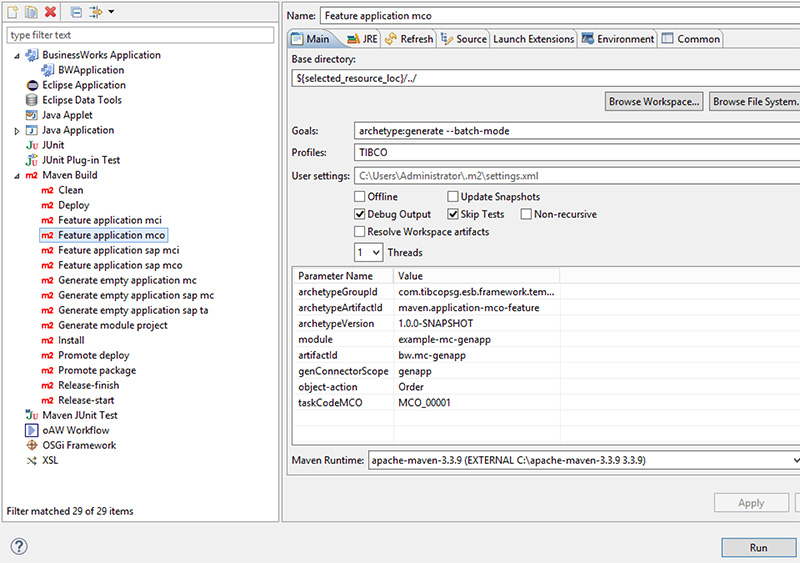
You have now added a logical connector MCI Service.

#### Add a Connector MCO service

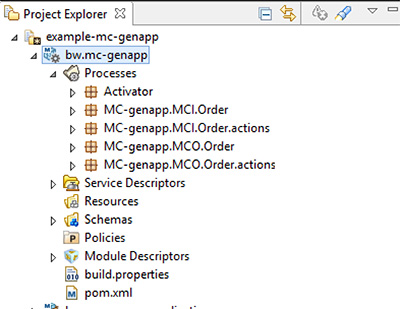
If you need to add an MCO connector service, please follow instructions in this section.

**You will now add logical connector MCO service in an existing Connector application.**

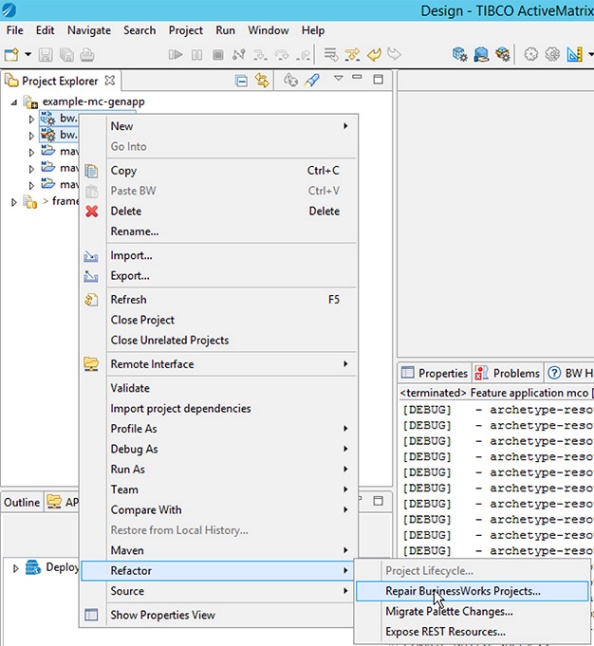
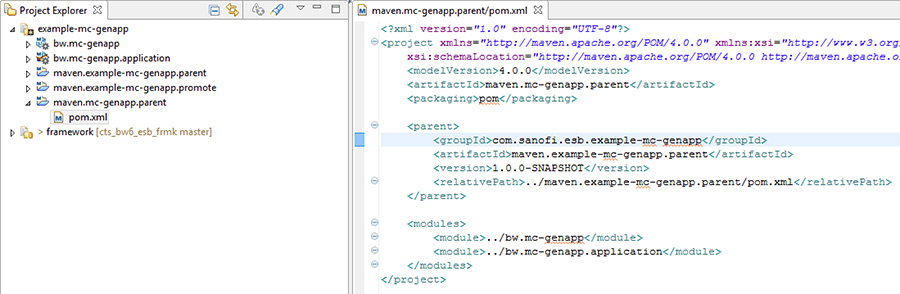
* Select the application module that you previously generated
* In the Maven builds, select the “Feature application mco”, and fill the values of these fields:
  + module
  + artifactId
  + genConnectorScope
  + object-action
  + taskCodeMCO



* Select the project, and refresh it (“F5” on your keyboard)



* Finish to refresh the project by right clicking and:
  + Refactor
  + Repair BusinessWorks Projects…



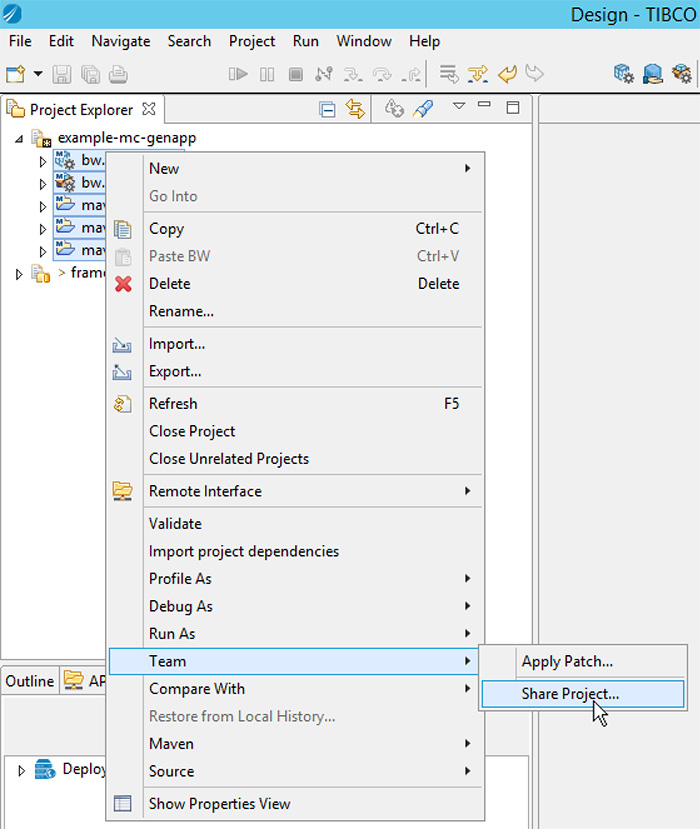
You have now added a logical connector MCO Service.

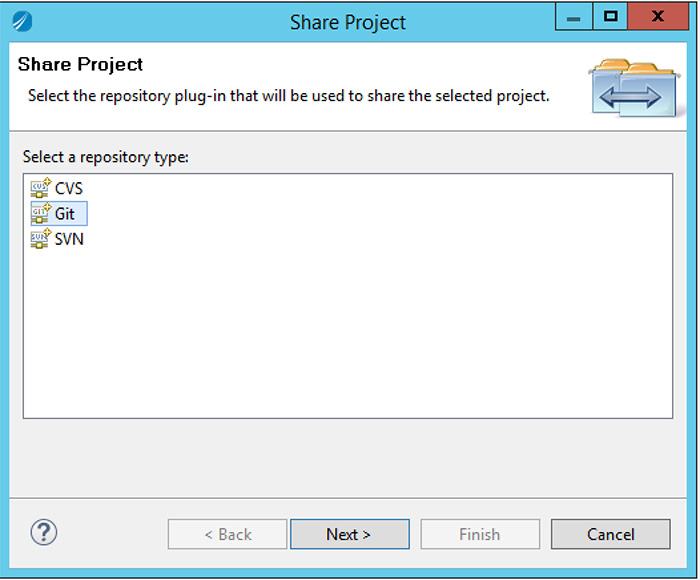
#### Add your project into the GIT Repository

If you have generated a new Connector application, then follow the instructions in this section.

If you only have generated new operations in an existing connector, you may simply commit your work.

**You will now add the newly generated Connector application projects into Git repository.**





* Select your local Git repository

You have now committed the Connector projects in Git.

## Template [SAP Middleware Connector]

### Application Design

The [SAP Middleware Connector] templates are used to generate SAP “MC” (Middleware Connector) TIBCO BusinessWorks 6.x application and services in respect with the SANOFI standard ESB framework design policies. SAP integration makes use of the TIBCO SAP plugin for BusinessWorks 6.x. The SAP Connector will typically be designed in two layers:

* Technical Adapter Layer (TA): contains a SAP Connection and an iDoc publisher service (SAP to JMS raw IDoc publishing).
* Logical Connector Layer (MC): contains the MCI and MCI IDoc publication
  + MCI: Middleware Connector In: asynchronous IDoc integration from SAP application to the ESB.
  + MCO: Middleware Connector Out: asynchronous IDoc integration from the ESB to SAP application.
  + MCS: Middleware Connector Service: BAPI interface running invocation from ESB to SAP.

*Note [July 20th 2017]: MCS/BAPI interface is not yet in the SAP template.*

The below diagram summarizes the SAP Connector Architecture;



* The TA layer takes care of publishing ALL IDocs out of a predefined SAP Connection.
* The MC layer manages MCI/MCO/MCS services.
  + MCI services read and parse IDocs from JMS using a JMS Selector (part of the TIBCO SAP activity setup).
  + MCO services are somehow inheriting from an odd design: IDocs are sent to a TAI service via JMS queue for integration to SAP. But the actual TAI service process will be generated in the MC application, instead of the TA itself. The reason for the current design is to limit the number of updates in the TA shared module that is today independent of specific objects/services exchanged. See the below note regarding the design.

Note: *TIBCO Engineering is considering upgrading the TIBCO SAP Plugin with a potential iDoc writer activity going straight (from ESB) to SAP without JMS transport intermediate. We would consider upgrading to the new design below IF this change occurs* ***(change not committed yet on July 20th 2017 )****.*



### Application Example

An example of a generated SAP middleware connector application is described in section 5.2.

### Template Design

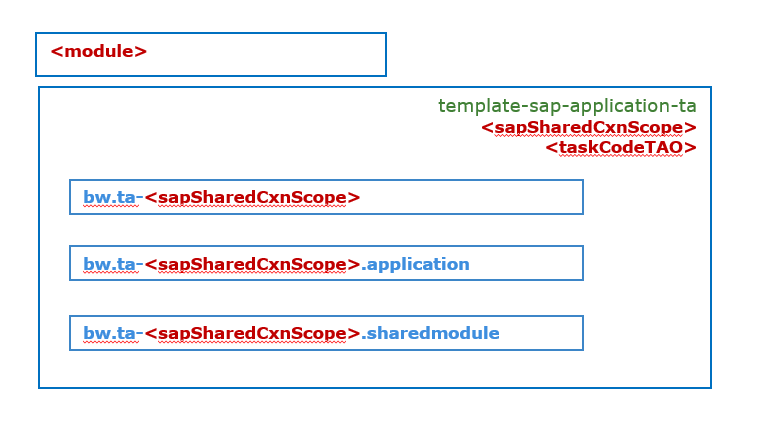
#### Architecture

The template distribution is based on a fully featured SAP connector application where special <keywords> are used for the replacement with variable parameters provided by the user. The parameters are automatically replaced by the template engine (maven) during the application generation. There are four options currently available in the [SAP Middleware Connector] application template:

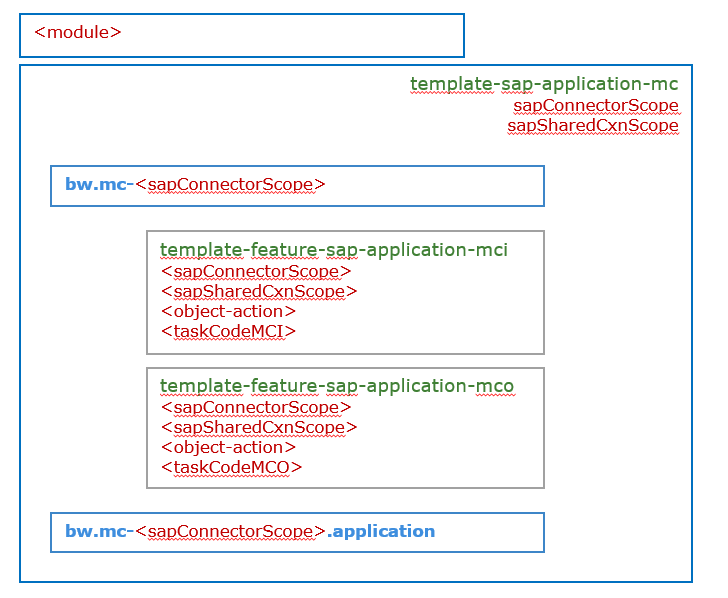
|  |  |  |
| --- | --- | --- |
| **Template SAP Middleware Connector** | | |
| **Function** | **Type** | **Description** |
| template-sap-application-ta | Template | Generates a new SAP Technical Adapter application. |
| template-sap-application-mc | Template | Generates a new SAP Logical Connector application. |
| template-feature-sap-application-mci | Template feature | Add an MCI operation to an existing SAP Logical Connector application. |
| template-feature-sap-application-mco | Template Feature | Add an MCO operation to an existing SAP Logical Connector application. |

The below diagram shows the template architecture and the variable parameters.

SAP Technical Adapter:



SAP Logical Connector and services



#### Module name

The “module name” is the name of the DEVOPS “module” the connector is to be member of. A [module] is just a top level folder containing one or many applications to be developed and released together in the same stream/visioning. You find the module folder in the GIT repository immediately under the “source” folder.

A module name can be defined as:

[ModuleName]: low-case string defining the overall functionality involved in the embedded applications.

If the module is designed to contain ONLY one application, you can give the module name the same name as the intended application name (see next section).

#### Technical Adapter name

The SAP Technical adapter is designed to manage one specific connection for a given scope of activity. The SAP connection will be hosted in a TIBCO BusinessWorks “sharedmodule”, for re-use by other logical connectors applications.

The Technical adapter application can be named based on the integrated SAP application and some optional sub-scopes qualifying the category of data or transaction planned for this connection.

The [TechnicalAdapterName] will be used in many items naming in the subsequent TIBCO BW6.x project content.

[TechnicalAdapterName]: ta-<integrated application name>(-<function sub-scope>)\*

Example:

* S4-FIN-masterdata: TA adapter for application SAP “S4”, domain finance, and masterdata data type.

#### Logical Connector name

The application name (or here “connector name”) is based on the following naming standard. The naming allows for defining application connector name based on the integrated application name and the variable scope of the service(s) to be implemented in the connector. The [LogicalConnectorName] will be used in many items naming in the subsequent TIBCO BW6.x project content.

[LogicalConnectorName]: mc-<integrated application name>(-<function sub-scope>)\*

<integrated application name>: name of the application the connector is designed for; example “S4” (for SAP 4 HANA)

<function sub-scope>: optional specific -businessDomain, -objectType, -objectName, -operation, defining a specific scope for the services in this connector.

Example of SAP connector scope:

* mc-S4: connector for running all integration services for the SAP instance S4.
* mc-S4-FIN: connector for SAP application S4, domain FINANCE.
* mc- S4-FIN -masterdata: connector for SAP application S4, domain FINANCE, datatype masterdata.
* mc- S4-FIN -masterdata-matmas: connector for SAP application S4, domain FINANCE, datatype masterdata, object matmas.
* mc- S4-FIN-masterdata-matmas-new: connector for SAP application S4, domain FINANCE, datatype masterdata, object matmas, operation new only.

#### Template properties.

The table below summarizes the properties to input for each template instance. Properties that are not displayed in the table

*Note: currently, the template configuration exposes the low level parameters required for the maven based generation. Later enhancement using eclipse plugin/form will allow more abstraction.*

Please, refer to the [ModuleName], [TechnicalAdapterName] and [LogicalConnectorName] defined in the above section 4.2.3.2, 4.2.3.3, 4.2.3.4 to use in the various properties below.

|  |  |  |
| --- | --- | --- |
| **Template Generic Middleware Connector properties and naming standard** | | |
| **Function** | **Template Property** | **Description** |
| template-sap-application-ta | module | [ModuleName] |
|  | artifactId | ta-[TechnicalAdapterName] |
|  | sapSharedCxnScope | [TechnicalAdapterName] |
|  | taskCodeTAO | This qualifies the technical adapter task, and should be provided in the specification. The taskCodeTAO naming is: TAO-nnnnn |
|  | appModuleName | ta-[TechnicalAdapterName] |
| template-sap-application-mc | module | [ModuleName] |
|  | artifactId | mc-[LogicalConnectorName] |
|  | sapConnectorScope | [LogicalConnectorName] |
|  | sapSharedCxnScope | [TechnicalAdapterName] |
|  | appModuleName | mc-[LogicalConnectorName] |
| template-feature-sap-application-mci | module | [ModuleName] |
|  | artifactId | bw.mc-[LogicalConnectorName] |
|  | sapConnectorScope | [LogicalConnectorName] |
|  | sapSharedCxnScope | [TechnicalAdapterName] |
|  | object-action | Name of the object and optional operation describing the MCI service to add, as defined in specification.  Use naming: <object>(-sub-action)\* |
|  | taskCodeMCI | Referenced MCI task code as provided in specification.  Naming: MCI-nnnnn |
| template-feature-sap-application-mco | module | [ModuleName] |
|  | artifactId | bw.mc-[LogicalConnectorName] |
|  | sapConnectorScope | [LogicalConnectorName] |
|  | sapSharedCxnScope | [TechnicalAdapterName] |
|  | object-action | Name of the object and optional operation describing the MCI service to add, as defined in specification.  Use naming: <object>(-sub-action)\* |
|  | taskCodeMCO | Referenced MCO task code as provided in specification.  Naming: MCO-nnnnn |
|  | taskCodeTAI | This qualifies the technical adapter task, and should be provided in the specification. The taskCodeTAI naming is: TAI-nnnnn |

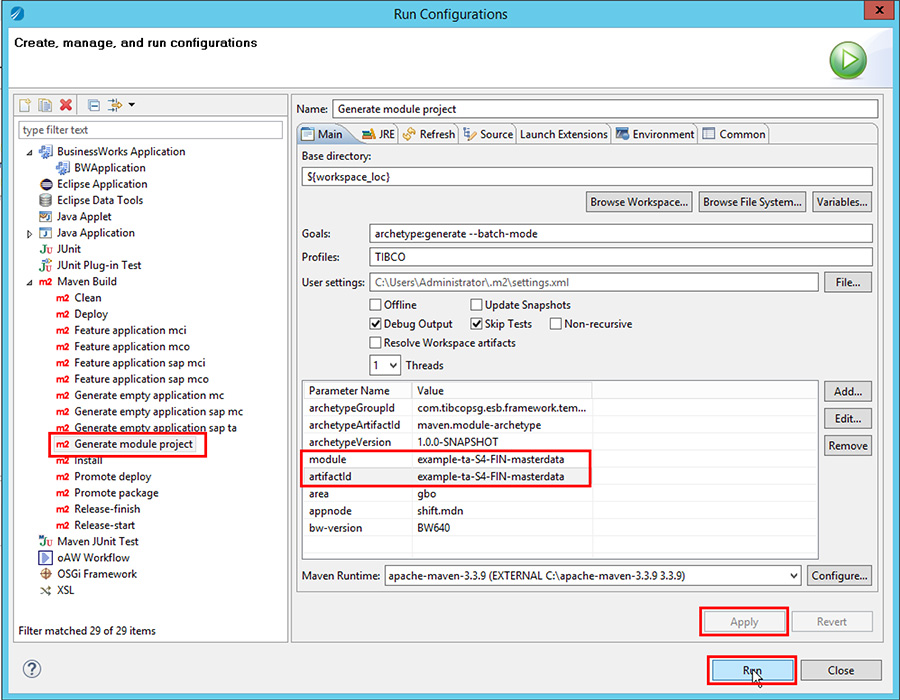
### Template usage

#### SAP Technical Adapter application

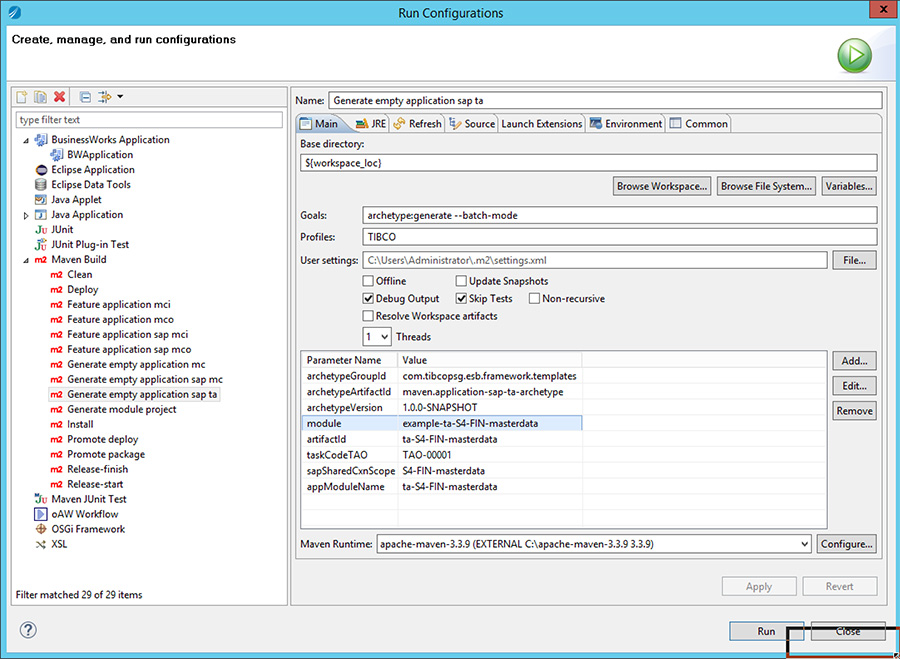
If it does not already exist, as required by your specification, you may generate a new SAP Technical Adapter application and shared module using the steps in this section.

**You will now generate a SAP Technical Adaptor.**

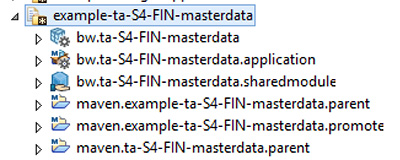
* First, generate a new module project



* Generate the technical adaptor, by selecting “Generating empty application sap ta” Maven build.
* Update the fields values:
  + module
  + artifactId
  + taskCodeTAO
  + sapSharedCxnScope
  + appModuleName



* Add a new working set with the name of the module
* Import the generated projects and add them to the new working set



* Don’t forget to update the pom.xml of the parent projects.
  + Maven parent project:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<artifactId>maven.ta-S4-FIN-masterdata.parent</artifactId>

<packaging>pom</packaging>

<parent>

<groupId>com.sanofi.esb.example-ta-S4-FIN-masterdata</groupId>

<artifactId>maven.example-ta-S4-FIN-masterdata.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../maven.example-ta-S4-FIN-masterdata.parent/pom.xml</relativePath>

</parent>

<modules>

<module>../bw.ta-S4-FIN-masterdata.sharedmodule</module>

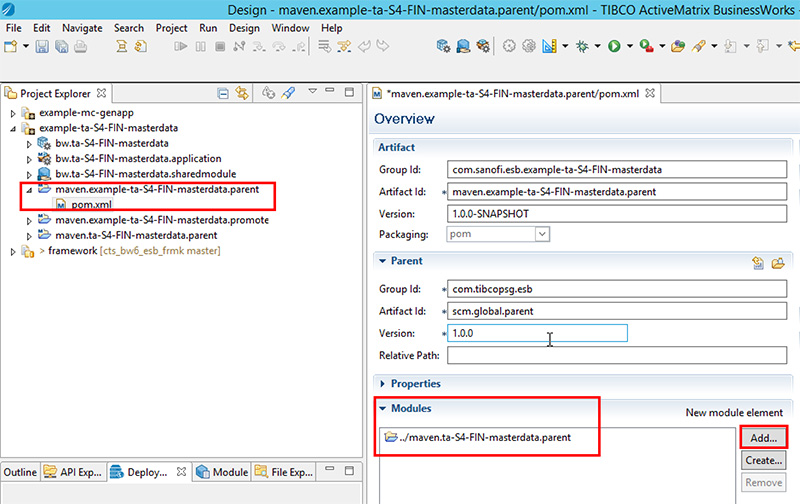
<module>../bw.ta-S4-FIN-masterdata</module>

<module>../bw.ta-S4-FIN-masterdata.application</module>

</modules>

</project>

* + Maven module parent project:

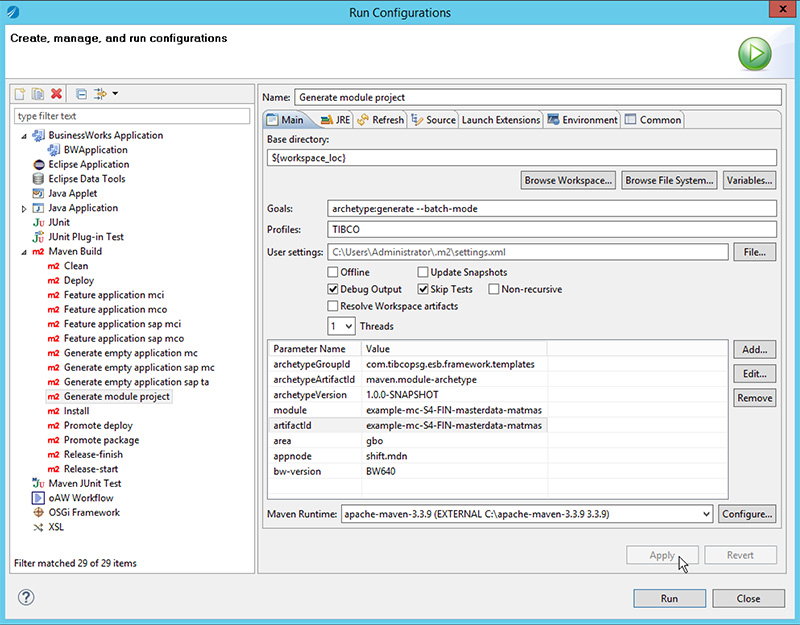


#### SAP Logical Connector application

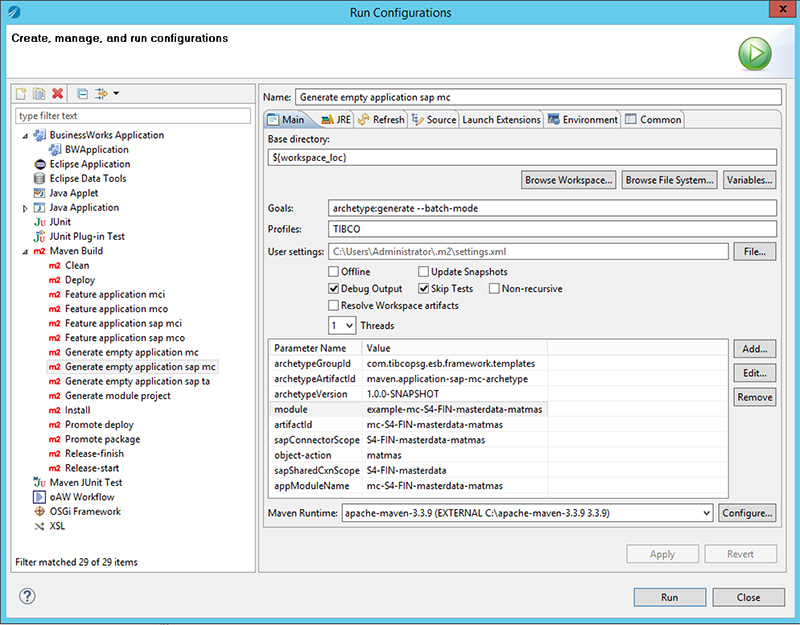
As required by your specification, if it does not already exist, you can follow the below instructions to generate a new SAP Logical Connector application. For this, you will need to have an existing SAP Technical adapter shared module to reference in order to manage references to the relevant SAP Connection.

**You will now generate a SAP logical connector application.**

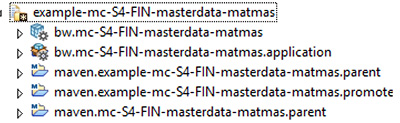
* First, generate a new module project



* Generate the SAP logical connector, by selecting “Generate empty application sap mc” Maven build.
* Update the fields values:
  + module
  + artifactId
  + sapConnectorScope
  + object-action
  + sapSharedCxnScope
  + appModuleName



* Add a new working set with the name of the module
* Import the generated projects and add them to the new working set



* Don’t forget to update the pom.xml of the parent projects.
  + Maven parent project:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<artifactId>maven.mc-S4-FIN-masterdata-matmas.parent</artifactId>

<packaging>pom</packaging>

<parent>

<groupId>com.sanofi.esb.example-mc-S4-FIN-masterdata-matmas</groupId>

<artifactId>maven.example-mc-S4-FIN-masterdata-matmas.parent</artifactId>

<version>1.0.0-SNAPSHOT</version>

<relativePath>../maven.example-mc-S4-FIN-masterdata-matmas.parent/pom.xml</relativePath>

</parent>

<modules>

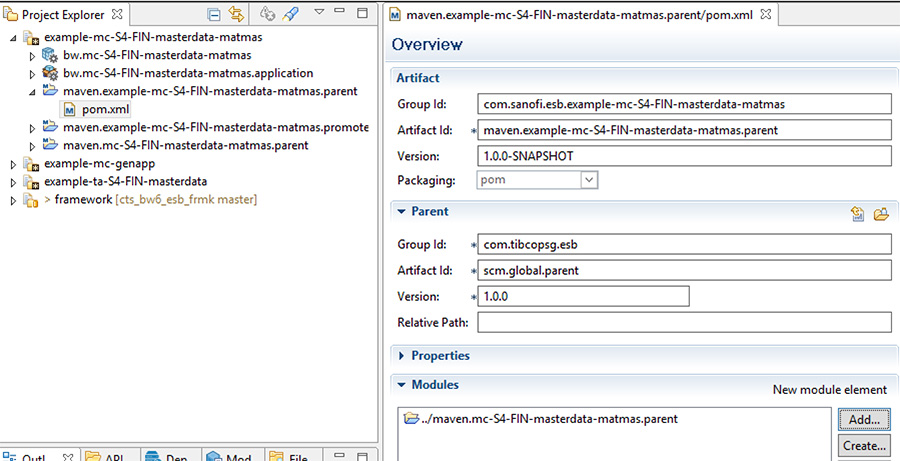
<module>../bw.mc-S4-FIN-masterdata-matmas</module>

<module>../bw.mc-S4-FIN-masterdata-matmas.application</module>

</modules>

</project>

* + Maven module parent project:



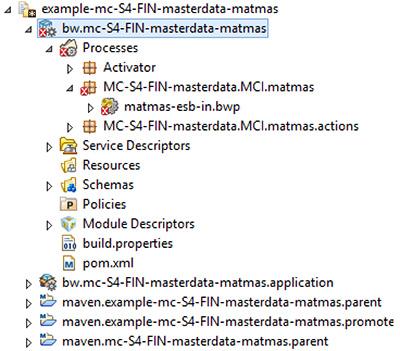
You have now generated a SAP logical connector.

#### Add a new SAP Logical Connector MCI Service

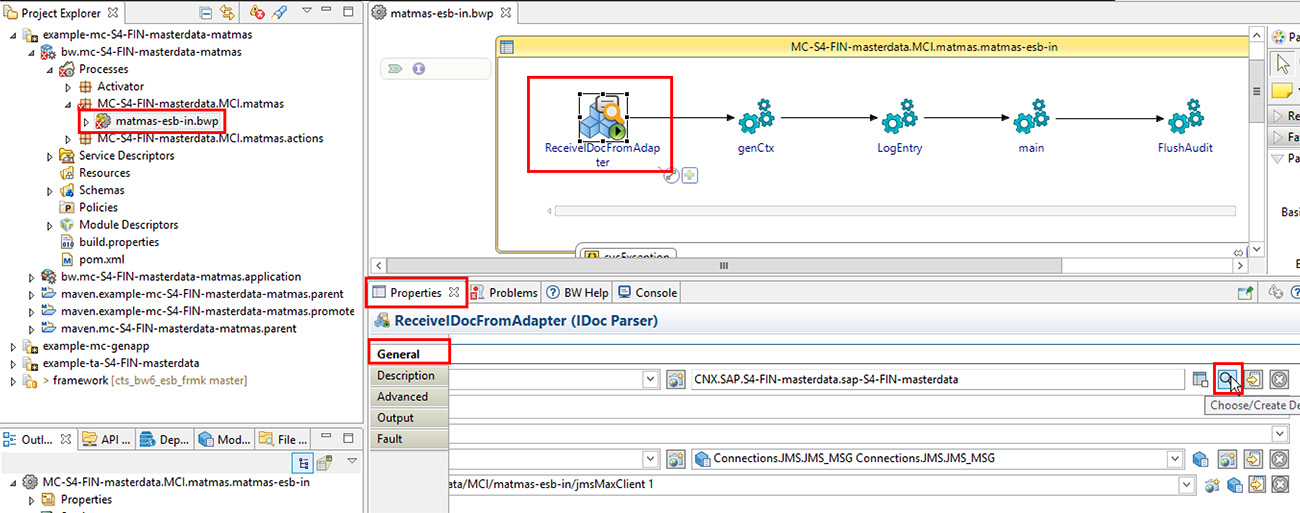
This section describes the steps to add a MCI service (Read IDoc from SAP and publish to ESB). For this you need an existing SAP Logical Connector application.

**You will now add a SAP logical connector IN on the existing logical project.**

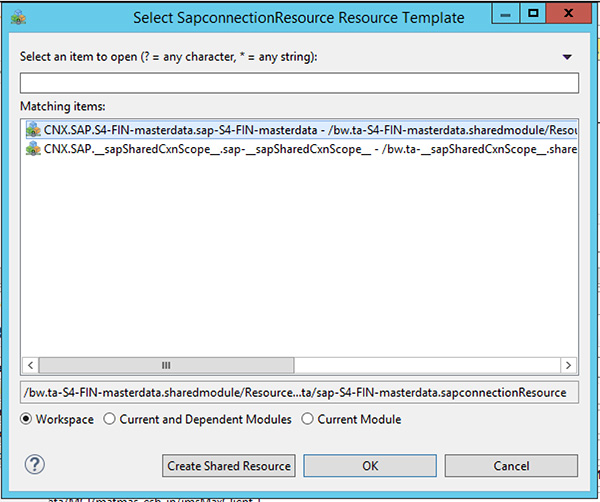
* Select the application module that you previously generated and select “Feature application sap mci” Maven build
* Update the fields values:
  + module
  + artifactId
  + sapConnectorScope
  + object-action
  + taskCodeMCI
  + sapSharedCxnScope
* Select the project, and refresh it (“F5” on your keyboard)
* Finish to refresh the project by right clicking and:
  + Refactor
  + Repair BusinessWorks Projects…
* A process is in error, open it:



* Select the IDoc Parser activity, in Properties tab, select General and click on the Loop icon:



* In the pop-up select the SAP connection resource of the technical adaptor:



You have added a SAP logical connector IN on the existing logical project.

#### Add a new SAP Logical Connector MCO Service

This section describes the steps to add a MCO service (Read object from ESB and publish IDoc to SAP). For this you need an existing SAP Logical Connector application.

**You will now add a SAP logical connector OUT on the existing logical project.**

* Select the application module that you previously generated and select “Feature application sap mco” Maven build
* Update the fields values:
  + module
  + artifactId
  + sapConnectorScope
  + object-action
  + taskCodeMCO
  + sapSharedCxnScope
* Select the project, and refresh it (“F5” on your keyboard)
* Finish to refresh the project by right clicking and:
  + Refactor
  + Repair BusinessWorks Projects…

You have added a SAP logical connector IN on the existing logical project.

# Application examples

## Example – Generic Middleware Connector

### Introduction

The [example-mc-genapp] application is an example of a [Generic Middleware Connector] application as issued from the application template described in section 4.1.

### Prerequisites

See prerequisites in section 3.

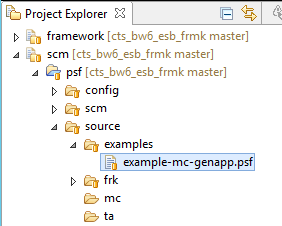
### Distribution

|  |  |
| --- | --- |
| **Application [example-mc-genapp]** | |
| Module | example-mc-genapp |
| PSF Source | /scm/psf/sources/examples/example-mc-genapp.psf |
| PSF Configuration (zone fr, env dev) | /scm/psf/config/modules/examples/config-fr-example-mc-genapp.psf |

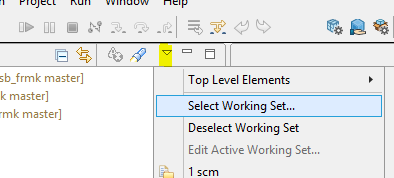
#### Source

**Load the [example-mc-application] source distribution**

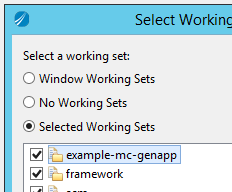
* In Studio, select the “/scm/psf/sources/examples/example-mc-genapp.psf” file from your PSF distribution. Right click and select “Import Project Set”.

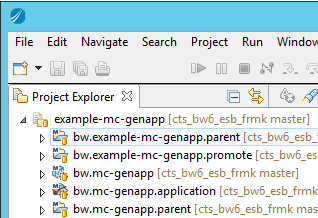


* Go to the view menu, then “Select Working Set”



* Select the [example-mc-genapp] working set.



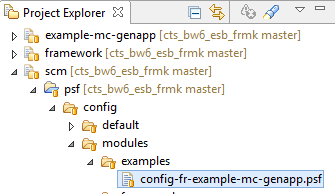


You have successfully loaded the [example-mc-genapp] working set in TIBCO Studio.

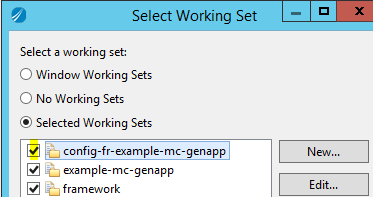
#### Configuration

**Load the [example-mc-application] configuration distribution, for the “fr” zone, and the “dev” environment**

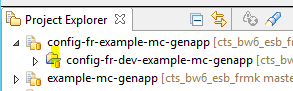
* Access the file “/scm/psf/config/modules/examples/config-fr-example-mc-genapp.psf”. Right click and select “import Project Set”.



* In the “view menu”, check the “config-fr-example-mc-genapp” Working Set.



* This displays the [example-mc-genapp] module configuration projects for all environments in the zone [fr].
* Inside this working set, you will find the specific configuration for the zone “fr”, and the environment “dev”.

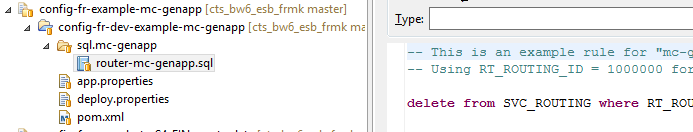


### Routing setup

The [example-mc-genapp] module includes an application [bw.mc-genapp] running an middleware connector example with an MCI and MCO service. A rule must be set in the Router database in order to dispatch the message from the MCI service to the MCO appropriate target JMS queue.

**Load the router rule for the [example-mc-genapp] application**

* Locate the configuration working set : “/config-fr-example-mc-genapp”
  + Locate the configuration project for the dev environment: config-fr-**dev**-example-mc-genapp
    - Locate the “sql.mc-genapp/**router-mc-genapp.sql**”



The sql file contains the script to record the router rule into the router database. This example rule may have been already loaded into the dev database.

* You may access the dev database to check if the rule exists.

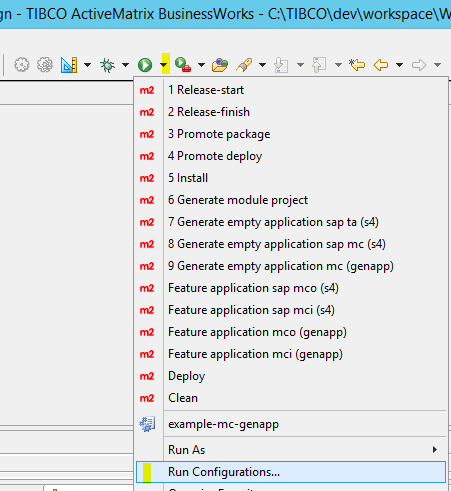
Select \* from SVC\_ROUTING where AP\_SRC\_APPLICATION =’mc-genapp’

* If not found you may run the “**router-mc-genapp.sql**” script to generate the rule.

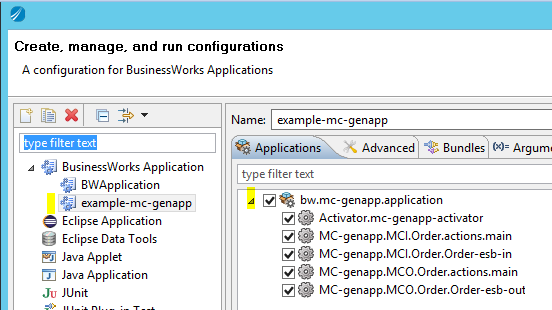
You have ensured the [mc-genapp] application had its router rule properly registered in the router database.

### Run the [example-mc-genapp] example

* In studio, select the run configuration



* Create a new run configuration named “example-mc-genapp”. Then select all projects.



* Select apply, then Run
* The project executes, and you should see the following engine output lines (extract);

...[AUDIT\_POLICY] - ...<tns:processPath>Activator.mc-genapp-activator</tns:processPath><tns:role>ENTRY</tns:role></tns:log></tns:logEventRecord>...

...[AUDIT\_POLICY] - ...<ns3:processPath>Activator.mc-genapp-activator</ns3:processPath><ns3:role>EXIT</ns3:role></ns3:log></ns3:logEventRecord>...

...[AUDIT\_POLICY] - ...<tns:processPath>MC-genapp.MCI.Order.Order-esb-in</tns:processPath><tns:role>ENTRY</tns:role></tns:log></tns:logEventRecord>...

...[AUDIT\_POLICY] - ...<ns4:processPath>RoutePublish.Action.SendOneWay</ns4:processPath><ns4:role>INFO</ns4:role><ns4:txt>ONEWAY message has been sent to target: [SANOFI.MCO.genapp.Order.esb-out]</ns4:txt><ns4:bodyTxt /><ns4:customStep>ROUTING\_DONE</ns4:customStep></ns4:log></ns4:logEventRecord>...

...[AUDIT\_POLICY] - ...<ns3:processPath>MC-genapp.MCI.Order.Order-esb-in</ns3:processPath><ns3:role>EXIT</ns3:role></ns3:log></ns3:logEventRecord>...

...[AUDIT\_POLICY] - ...<tns:processPath>MC-genapp.MCO.Order.Order-esb-out</tns:processPath><tns:role>ENTRY</tns:role></tns:log></tns:logEventRecord>...

...[AUDIT\_POLICY] - ...<ns3:processPath>MC-genapp.MCO.Order.Order-esb-out</ns3:processPath><ns3:role>EXIT</ns3:role></ns3:log></ns3:logEventRecord>...

## Examples – SAP Middleware Connector

### Introduction

The design of SAP Connector application is described in the template section **Error! Reference source not found.**. A SAP Connector comes in at least two modules.

* The [example-ta-S4-FIN-masterdata] application is an example of a Technical adapter (TA) for integration with SAP. It includes a TIBCO BW6.x shared module containing a SAP connection resource as well as an application module implementing the raw IDoc publication from SAP to a JMS Queue, for consumption by the logical connector layer (MC, for Middleware Connector).
* The [example-ta-S4-FIN-masterdata-matmass] application is an example of a logical middleware connector [MC] supporting MCI and MCO services. Such a SAP application connector relies on a dependency on a SAP TA adapter application, such as the [example-ta-S4-FIN-masterdata] application for the purpose of the example.

### Prerequisites

See general prerequisites in section 3.

You will also need an access to an SAP S4 instance ready for middleware connector integration testing. Check with SANOFI projects lead to obtain all the SAP connectivity details to prepare your IDoc IN/OUT messaging scenario.

Collect the values for the variables listed in the below table, that corresponds to the TIBCO Technical adapter shared resource SAP connectivity properties. Below values are provides as an example only. Grey area indicates the parameter is currently not in use.

|  |  |
| --- | --- |
| **SAP Connectivity parameters** | |
| **Property** | **Value** |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPAppServer |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSystemNumber |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPClient | 800 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPUserName | sapadapter5 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPPassword | \*\*\*\*\*\*\*\* |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSystemName | SE1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPMsgServer | 10.108.80.239 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPLogonGroup | PUBLIC |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPProgramID | tibcosappid5 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSrvCnxGatewayService | sapgw00 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSrvCnxGatewayHost | 10.108.80.239 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPmaxReconnectAttempt | -1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPDelayReconnectAttemptMS | 30000 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPMaxConnections | 1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPRFCTrace | 1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPUseSAPGUI | 0 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBUrl | na |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDLoginTimeoutMS | 3000 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBMaxCxn | 1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBRetryCount | 3 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBRetryDelayMS | 0 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBUser | user |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPTIDDBPassword | \*\*\*\*\* |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPCodePage |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSrvCnxStartupAttemptDelayS | 60 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSrvCnxMaxConnection | 1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SrvCnxRFCTrace | 1 |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSncMode |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSncPartnername |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSncQop |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSncLib |  |
| /SharedConnections/SAP/S4-FIN-masterdata/SAPSrvCnxSncMyname |  |

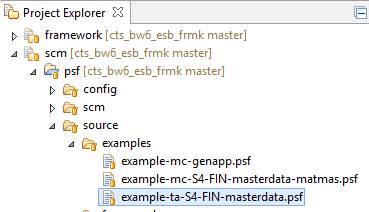
### Distribution

|  |  |
| --- | --- |
| **Application [example-ta-S4-FIN-masterdata]** | |
| Description | Technical adapter application for the scope **S4-FIN-masterdata** |
| Module | example-ta-S4-FIN-masterdata |
| PSF Source | /scm/psf/sources/examples/example-ta-S4-FIN-masterdata.psf |
| PSF Configuration (zone fr, env dev) | /scm/psf/config/modules/examples/config-fr-example-ta-S4-FIN-masterdata.psf |
| **Application [example-mc-S4-FIN-masterdata-matmas]** | |
| Description | Logical connector application for the scope **S4-FIN-masterdata-matmas** |
| Module | example-mc-S4-FIN-masterdata-matmas |
| PSF Source | /scm/psf/sources/examples/example-mc-S4-FIN-masterdata-matmas.psf |
| PSF Configuration (zone fr, env dev) | /scm/psf/config/modules/examples/config-fr-mc-S4-FIN-masterdata-matmas.psf |

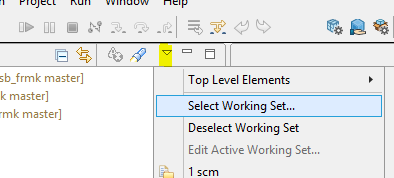
#### Source – SAP technical adapter

**Load the [example-ta-S4-FIN-masterdata] source distribution**

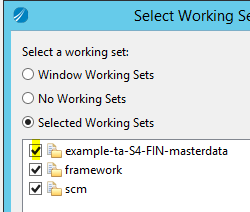
* In Studio, select the “/scm/psf/sources/examples/example-ta-S4-FIN-masterdata.psf” file from your PSF distribution. Right click and select “Import Project Set”.

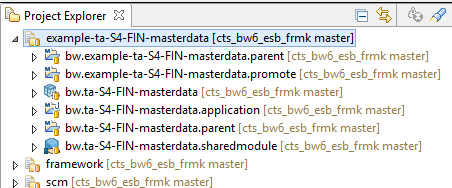


* Go to the view menu, then “Select Working Set”



* Select the [example-ta-S4-FIN-masterdata] working set.



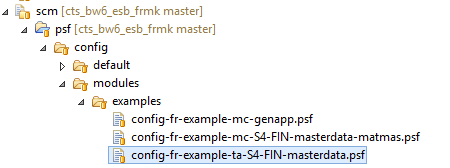


You have successfully loaded the [example-ta-S4-FIN-masterdata] working set in TIBCO Studio.

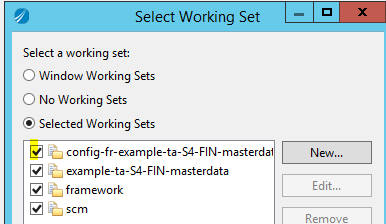
#### Configuration – SAP technical adapter

**Load the [example-ta-S4-FIN-masterdata] configuration distribution, for the “fr” zone, and the “dev” environment**

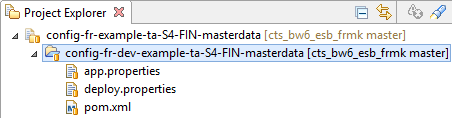
* Access the file “/scm/psf/config/modules/examples/config-fr-example-ta-S4-FIN-masterdata.psf”. Right click and select “import Project Set”.



* In the “view menu”, check the “config-fr-example-ta-S4-FIN-masterdata” Working Set.



* This displays the [example-ta-S4-FIN-masterdata] module configuration projects for all environments in the zone [fr].
* Inside this working set, you will find the specific configuration for the zone “fr”, and the environment “dev”. This configuration is required to deploy the snapshot in the dev environment, but is not mandatory for running the application in studio (where default module application properties would be used instead).

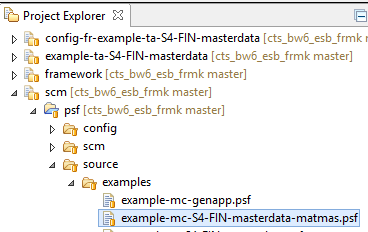


**You have loaded the [example-ta-S4-FIN-masterdata] configuration distribution, for the “fr” zone, and the “dev” environment.**

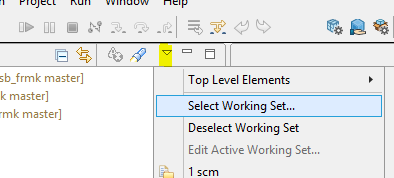
#### Source – SAP logical Connector

**Load the [example-mc-S4-FIN-masterdata-matmas] source distribution**

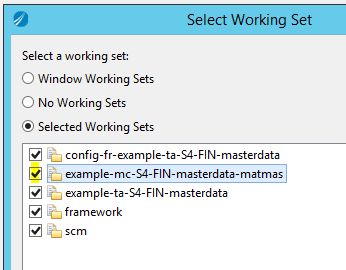
* In Studio, select the “/scm/psf/sources/examples/example-mc-S4-FIN-masterdata-matmas.psf” file from your PSF distribution. Right click and select “Import Project Set”.

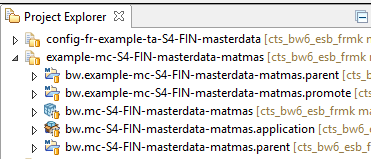


* Go to the view menu, then “Select Working Set”



* Select the [example-mc-S4-FIN-masterdata-matmas] working set.



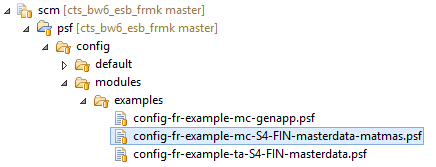


You have successfully loaded the [example-mc-S4-FIN-masterdata-matmas] working set in TIBCO Studio.

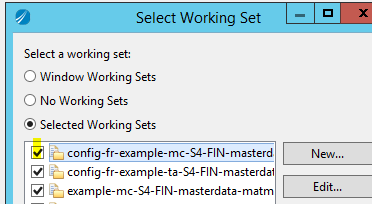
#### Configuration – SAP logical Connector

**Load the [example-mc-S4-FIN-masterdata-matmas] configuration distribution, for the “fr” zone, and the “dev” environment**

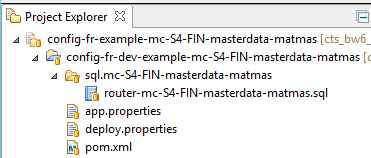
* Access the file “/scm/psf/config/modules/examples/config-fr-example-mc-S4-FIN-masterdata-matmas.psf”. Right click and select “import Project Set”.



* In the “view menu”, check the “config-fr-example-mc-S4-FIN-masterdata-matmas” Working Set.



* This displays the [example-mc-S4-FIN-masterdata-matmas] module configuration projects for all environments in the zone [fr].
* Inside this working set, you will find the specific configuration for the zone “fr”, and the environment “dev”. This configuration is required to deploy the snapshot in the dev environment or if you need to setup the router rules. However, it is not mandatory for running the application in studio (where default module application properties would be used instead), assuming router rule is already configured.



**You have loaded the [example-mc-S4-FIN-masterdata-matmas] configuration distribution, for the “fr” zone, and the “dev” environment.**

### Routing setup – SAP logical Connector

The [example-mc-S4-FIN-masterdata-matmas] module includes an application module [bw.mc-S4-FIN-masterdata-matmas] running a SAP connector example with an MCI (IDoc In) and MCO (IDoc out) service. A rule must be set in the Router database in order to dispatch the message from the MCI service to the MCO appropriate target JMS queue.

**Load the router rule for the [bw.mc-S4-FIN-masterdata-matmas] application module**

* Locate the configuration working set: “/config-fr-example-mc-S4-FIN-masterdata-matmas”
  + Locate the configuration project for the dev environment: config-**fr**-**dev**-example-mc-S4-FIN-masterdata-matmas
    - Locate the “sql.mc-S4-FIN-masterdata-matmas/**router-mc-S4-FIN-masterdata-matmas.sql**” script.



The sql file contains the script to record the router rule into the router database. This example rule may have been already loaded into the dev database.

* You may access the dev database to check if the rule exists.

Select \* from SVC\_ROUTING where AP\_SRC\_APPLICATION =mc-S4-FIN-masterdata-matmas’

* If not found you may run the “**router-mc-S4-FIN-masterdata-matmas.sql**” script to generate the rule.

You have ensured the [mc-S4-FIN-masterdata-matmas] application had its router rule properly registered in the router database.

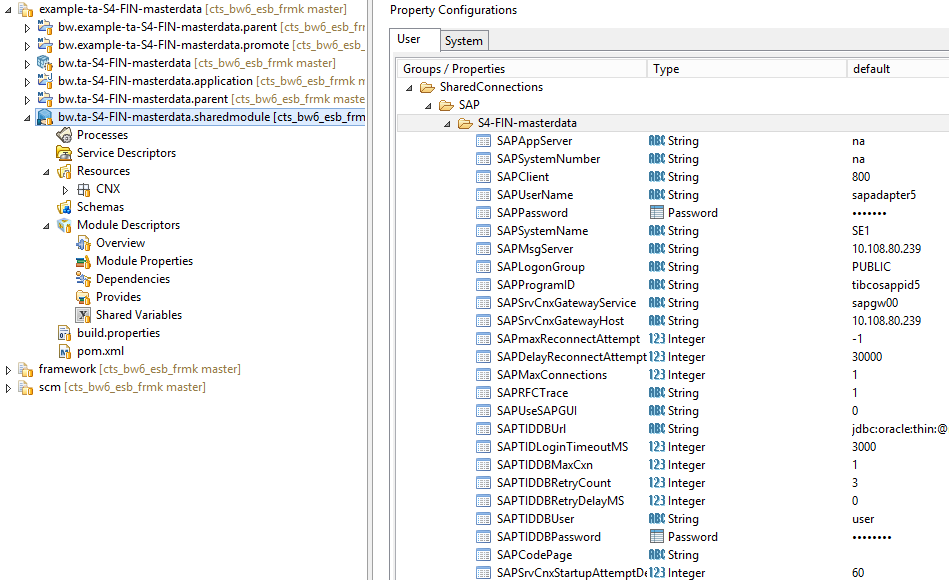
### Run the complete SAP Connector Application

#### Configure the Technical adapter SAP connectivity

**Configure SAP Connectivity**

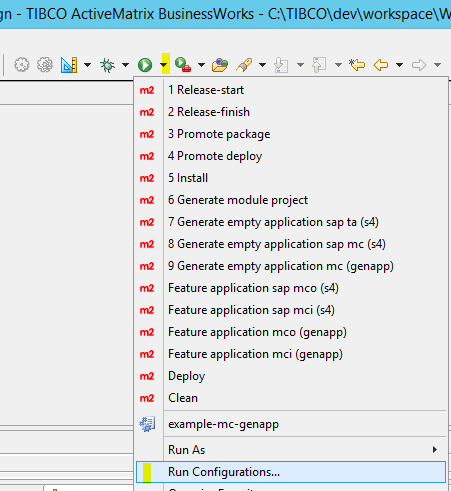
If not already defaulted to SANOFI sample test values, you will need to configure the SAP connectivity parameter in the SAP Technical adapter. For running the example in the TIBCO Studio, you may simply configure the variables in the Shared module properties; Check section 5.2.2 to list the required values with your SANOFI SAP Application expert.

Open the “bw.ta-S4-FIN-masterdata.sharedmodule/Module Descriptors/Module Properties” element to edit the SAP Connection parameters.

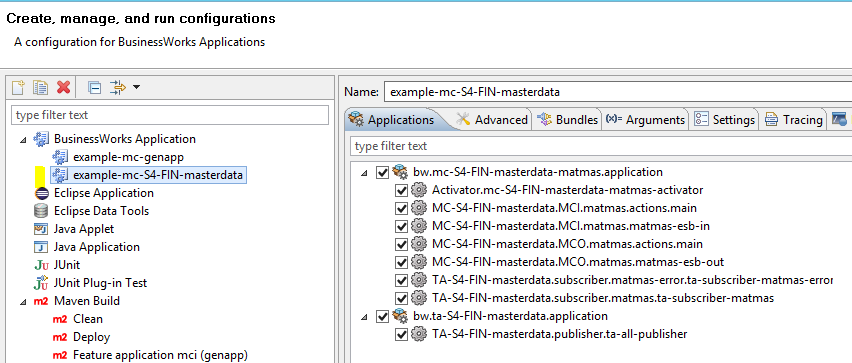


#### Run the connector

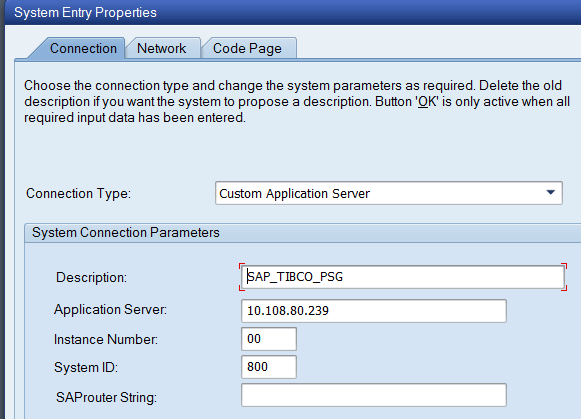
* In studio, select the run configuration



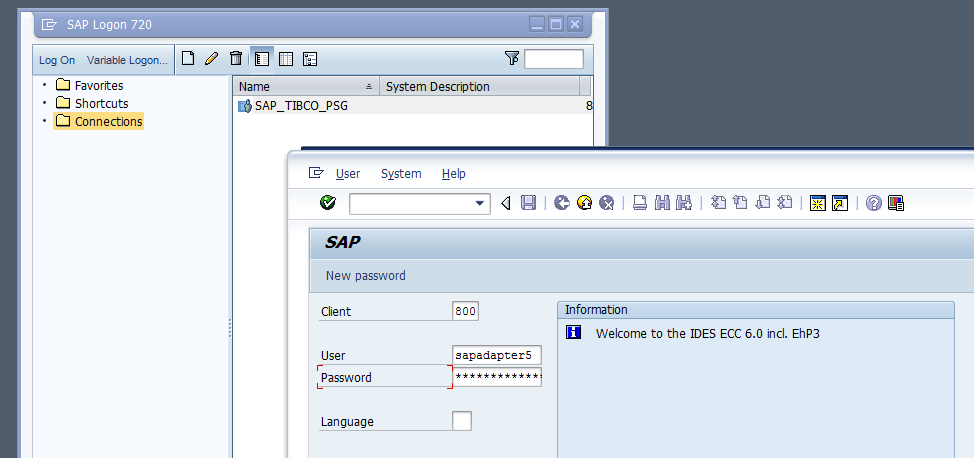
* Create a new run configuration named “example-mc-S4-FIN-masterdata”. Then select all projects.



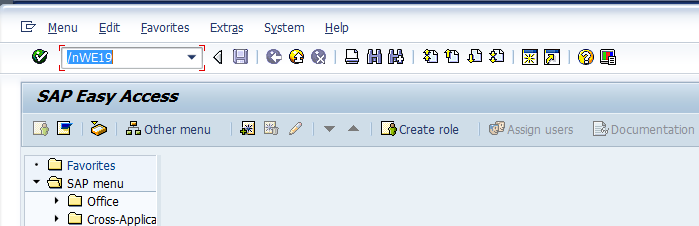
* Select apply, then Run
* In order to test your SAP Connector, you may use the SAP UI to copy an IDOC for publication; the following guideline **is provided as an example only**. This will depend on your SAP System.
* Run your SAP UI application. And configure the appropriate connection. Example:



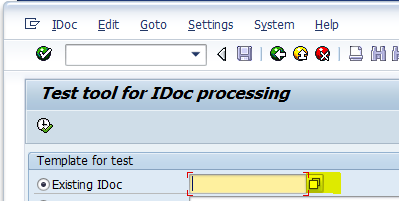
* Connect to SAP with SAP UI; Example:

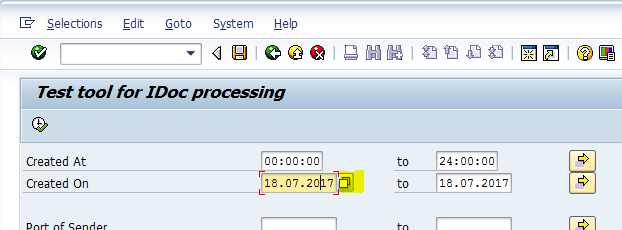


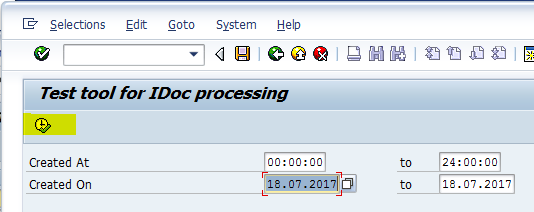
* Execute a WE19 command



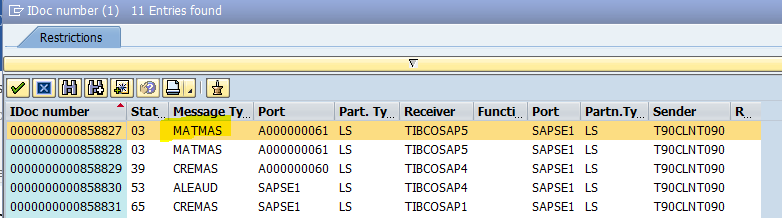
* Search for an appropriate MATMAS IDOC; you may select a date you know a relevant matmas IDoc was sent.



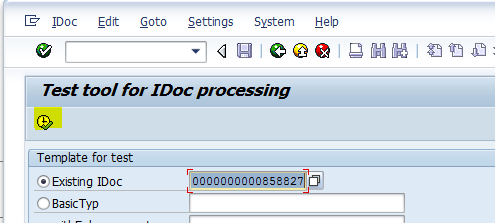




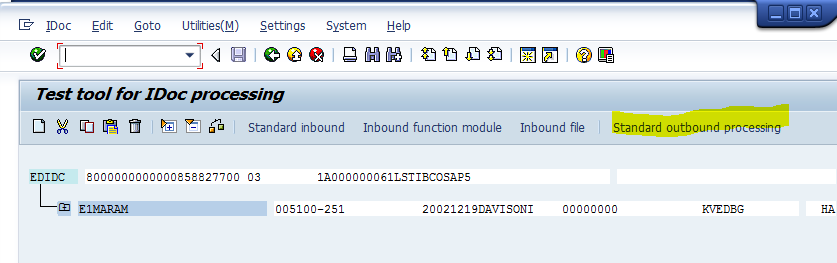
* Then select the relevant IDoc (double click)



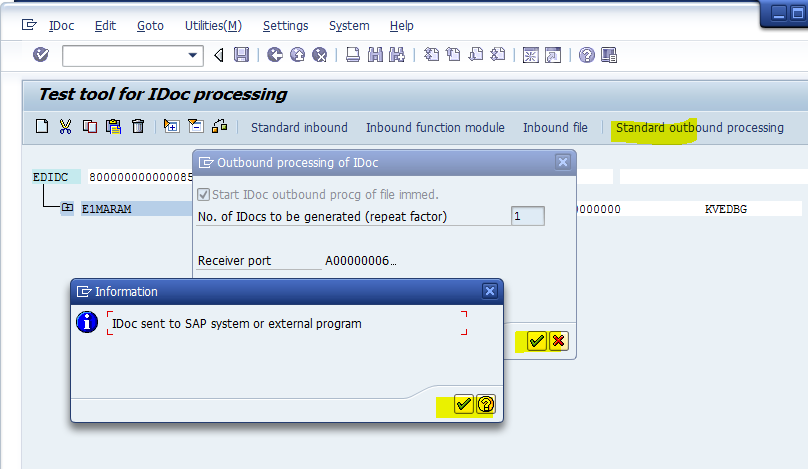
* Select it



* Select “Standard outbound processing”



* Dispatch the IDoc copy



* Assuming accurate parameters (connectivity, client, program ID …), this should trigger the scenario in TIBCO BusinessWorks from TA publisher, to MCI, then routing to MCO, then TA Subscriber.

...[AUDIT\_POLICY] ...<ns4:processPath><ns4:role>INFO</ns4:role><ns4:txt>ta-all-publisher sent idoc No [0000000000858838].</ns4:txt><ns4:bodyTxt /></ns4:log><ns4:appContext><ns4:nvpElement><ns4:name>SAP.IDocNumber</ns4:name><ns4:value>0000000000858838</ns4:value></ns4:nvpElement><ns4:nvpElement><ns4:name>SAP.in.TID</ns4:name><ns4:value>0A6C50EF655B596E3C870187</ns4:value></ns4:nvpElement></ns4:appContext></ns4:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><tns:role>ENTRY</tns:role></tns:log><tns:appContext><tns:nvpElement><tns:name>SAP.IDocNumber</tns:name><tns:value>0000000000858838</tns:value></tns:nvpElement></tns:appContext></tns:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><ns4:role>INFO</ns4:role><ns4:txt>ONEWAY message has been sent to target: [SANOFI.MCO.S4-FIN-masterdata.matmas.esb-out]</ns4:txt><ns4:bodyTxt /><ns4:customStep>ROUTING\_DONE</ns4:customStep></ns4:log></ns4:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><ns3:role>EXIT</ns3:role></ns3:log></ns4:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><tns:role>ENTRY</tns:role></tns:log></tns:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><ns3:role>EXIT</ns3:role></ns3:log></ns4:logEventRecord>...

...[AUDIT\_POLICY] ...<ns4:processPath><ns4:role>INFO</ns4:role><ns4:txt>ta-subscriber-matmas: Sent IDoc to SAP - MsgId [ID:EMS-SERVER.2F05966B17A59:13] TID [0A6C50EF655B596E3D230188]</ns4:txt><ns4:bodyTxt /></ns4:log><ns4:appContext><ns4:nvpElement><ns4:name>SAP.out.TID</ns4:name><ns4:value>0A6C50EF655B596E3D230188</ns4:value></ns4:nvpElement></ns4:appContext></ns4:logEventRecord>...