Consulting
Services

CO-OP Bank
ESB
How To: Create BW Services from Templates

Document Version: v1.1.0

Document Date: March 2018

http://www.TIBCO.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005

Copyright Notice

Trademarks

Confidentiality

Content Warranty

For more information, please contact:

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304
USA

Table of Contents

Table of Figures

Table of Tables

1 Document Control

1.1 Versioning Information

Issue Date	Version	Description	Author
06/01/2016	0.1.0	First draft – Based on reference document from Pierre Ayel	Julian Cranfield

1.2 Distribution Information

1.2.1 CO-OP Bank

Name	Project Role

1.2.2 TIBCO

Name	Project Role
Andrew Dearing	Project Delivery Manager

1.3 Reference Documents

Name	Author

1.4 Document Acceptance

Signed for and on behalf of The Automobile Association	
Signature of authorised officer:	
Name of authorised officer:	
Witness:	
Date:	

2 Introduction

To enable the standardisation of services within CO-OP Bank using TIBCO BusinessWorks 6, a few templates have been created as starting points for the creation of services.

These templates do not dictate who the service should operate or the orchestration of the activities within the service process, however, they do position audit, exception handing within the process orchestration in a standardised way.

The naming conventions in this document have been created to ensure a standard across all services and therefore a general understanding. For example, when trying to find the endpoint URL for a service in the module properties there will be a standard naming convention so this can be located quickly and accurately.

2.1 Scope

This document and the templates that it references enable the creation of business and technical services.

2.2 Purpose

The templates and the use of the templates as laid out in this document ensure the standardization of TIBCO BusinessWorks development for services.

2.3 Audience

All developers of services using TIBCO BusinessWorks should adhere to the practices in this document.

2.4 Acronyms

Table 1: Acronyms

Acronym	Definition
BST	TIBCO Business Studio: The Eclipse IDE for building BW applications
BW	TIBCO BusinessWorks
EMS	TIBCO Enterprise Messaging Service
ESB	Enterprise Service Bus, a technical capability providing runtime facilities for transformation and mapping, integration and process management (orchestration)
FTP	File Transfer Protocol
JDBC	Java API: generic API to integrate with databases
JMS	Java API: Java Messaging Service
JSON	Data format: Javascript Object Notation
HTTP	Network transport protocol: Hypertext Transport Protocol

Acronym	Definition
GIT	Git source control system
XML	Data format: Extended Markup Language

2.5 Issues

Table 2: Issues

#	Status	Owner	Description / Resolution

2.6 Risks

TBC

3 Introduction to the Templates

Business service – these are front-end requests that users see. Business services call to technical service(s) for action.

Technical service – these provide the undercover operations, like talking to a database, and they are unaware of the business services and business logic.

Application – applications are **deployable** which means there is either a SOAP service bound to JMS or HTTP, or REST over HTTP.

Shared module – these are **not deployable** since they have no bindings, and are used for inline calls only.

4 Create a Business Service Provider as an Application

This chapter explains how to create a new Business service exposed on SOAP over HTTP/JMS.

4.1 Template Overview

The template module APP_BSServiceTemplate.module allows you to create a new application for a Business service. It contains the following objects:

- Processes:
  - **BSService**: the service implementation.
  - **Activator**: the process executed when the application starts up inside an appnode. This can be used to trace configuration settings (such as back-end URLs) or call other Activator required in dependent modules.
- WSDLs:
  - **ModuleActivator.wsdl**: the abstract WSDL of the **Activator** process (this one should never be changed at all: it is created by BW when the Activator process is created).
- Resources:
  - **Server-BW01**: the HTTP server where the Business service will be exposed.
  - **JNDIClient-ESB01**: the JNDI connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.
  - **JMSClient-ESB01**: the JMS connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.

Notes:

1 The JNDI and JMS connections used by the System Error Handler and System Audit are separated from JNDIClient-ESB01 and JMSClient-ESB01. They are contained in the LIB_TSUTIL_SystemErrorHandler_Client and LIB_TSUTIL_SystemAudit_Client modules.

4.2 Pre-Requisites

The XSD and WSDL have been created with all the correct naming structures.

4.2.1 Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder APP_BSServiceTemplate.module into the folder **<GIT> / trunk / BW / BusinessServices**.
2. Rename the copied folder into “APP_BS<Area>_<ServiceName>”.

3. With a text editor, open the project file into the copied folder and change the project name at the top from APP_ServiceTemplate.module into APP_BS<Area>_<ServiceName>.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>APP_BSGEN_ServiceA</name>
	<comment></comment>
…
```

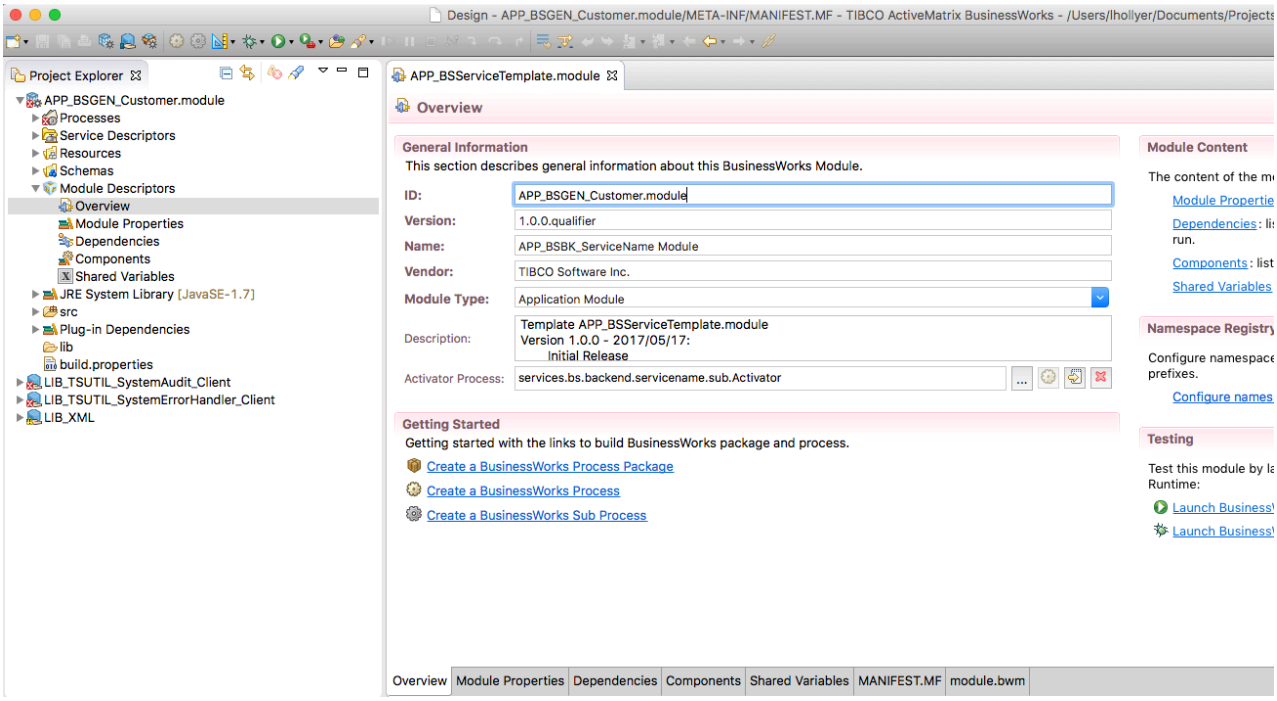
With TIBCO BusinessStudio

4. Start TIBCO BusinessStudio
5. In your workspace, import the following modules:
    a. LIB_XML (from <GIT> / trunk / XML)
    b. LIB_TSUTIL_SystemAudit_Client (from <GIT> / trunk / BW / TechnicalServices)
    c. LIB_TSUTIL_SystemErrorHandler_Client (from <GIT> / trunk / BW / TechnicalServices)
    d. APP_<TYPE><Area>_<ServiceName> (from <GIT> / trunk / BW / BusinessServices)

All subsequent changes are done on the APP_<TYPE><Area>_<ServiceName> module:

6. Go to the module overview.

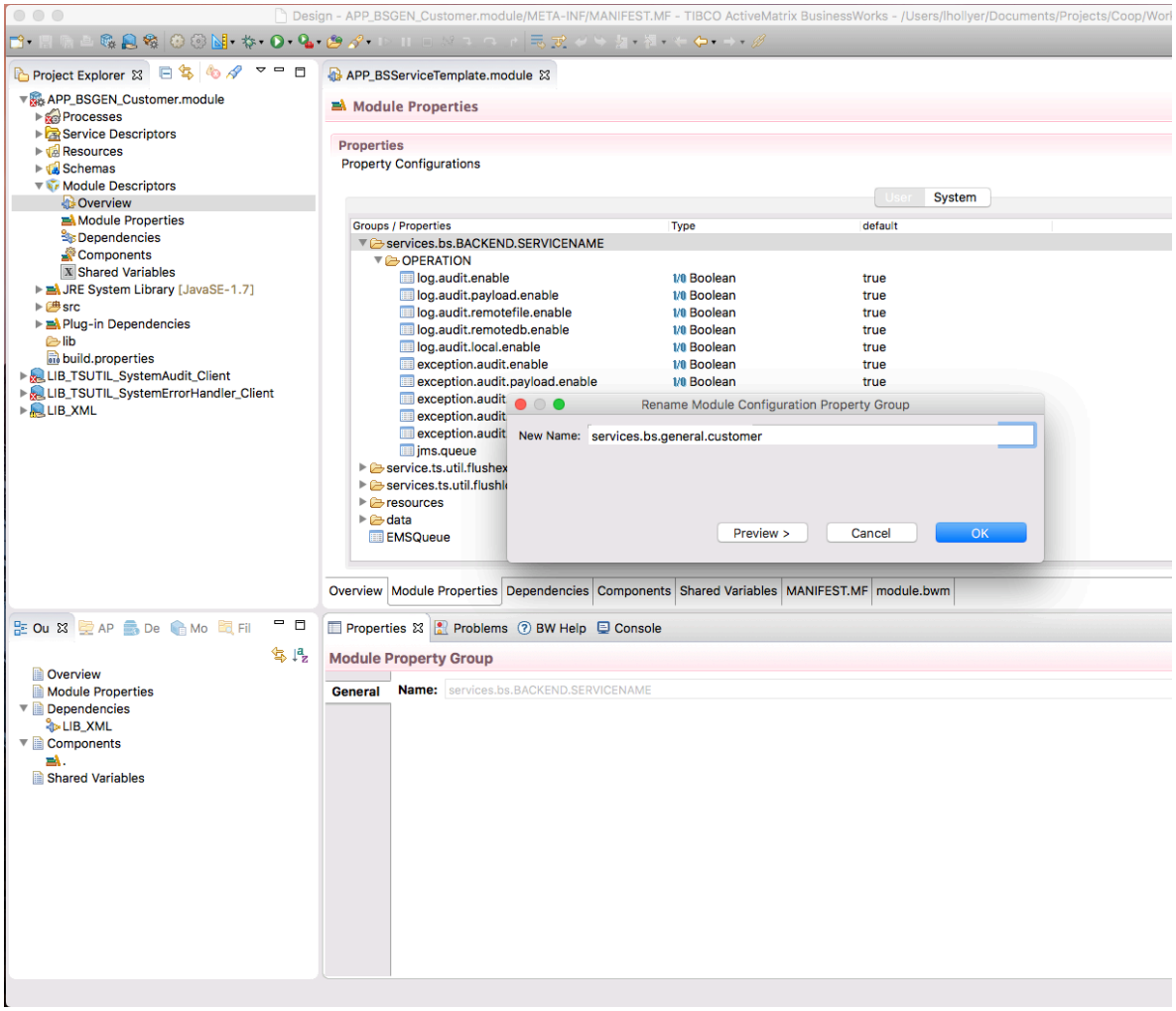
7. Change the name into “APP_BS<Area>_<ServiceName> Module”:



4.2.2 Change the Module Properties

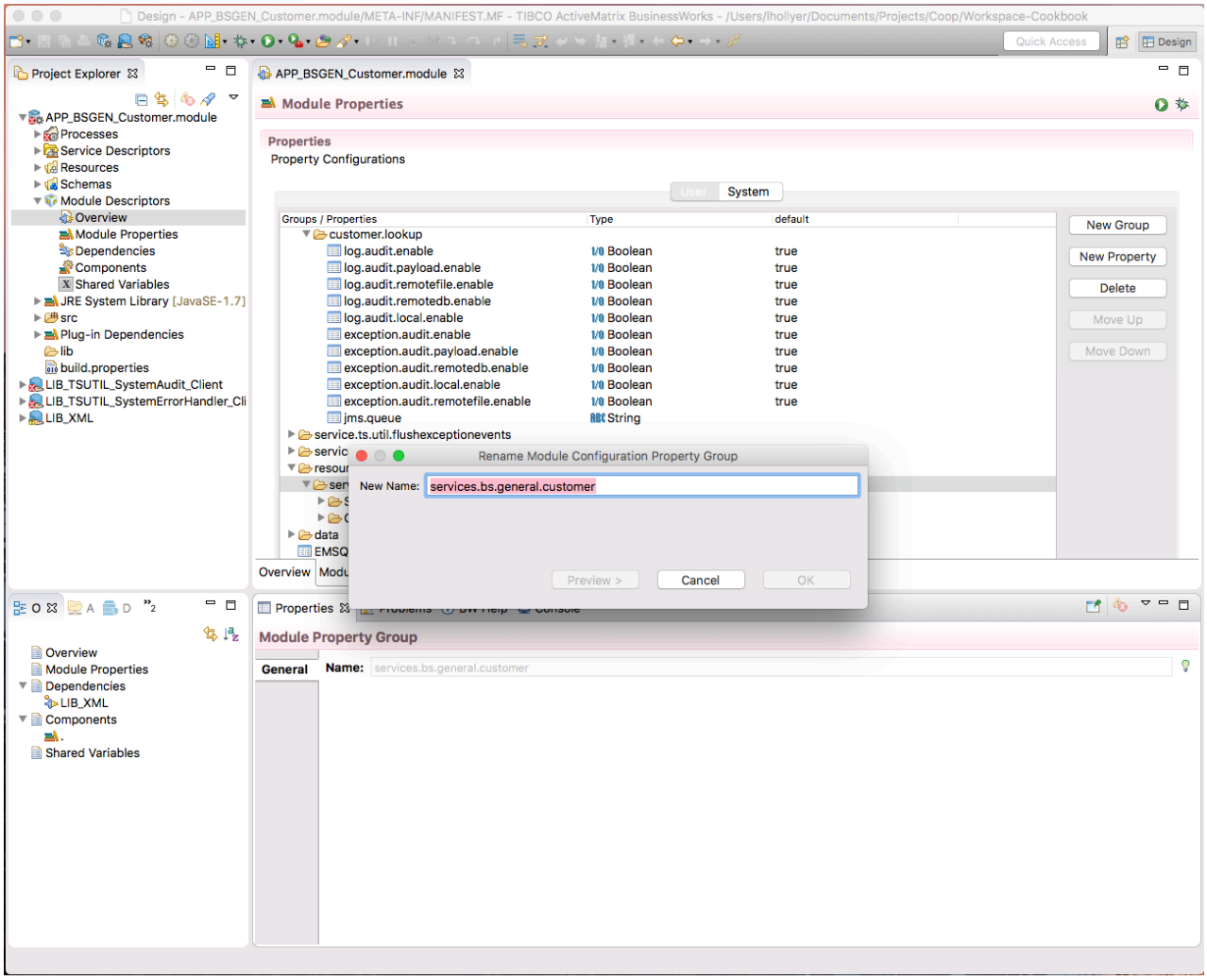
1. Go to the module properties.

2. Select the group “services.bs.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.<type><area>.<servicename>”:



3. In this group, use the same technique to rename the group “OPERATION” into “<ServiceName>”.

4. Select the group “resources / services.bs.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.<type><area>.<servicename>”:



5. In this group, change the value of the HTTP port to the port assigned for the service.

4.2.3 Rename the Resources

1. Open the HTTP Server resource. In the properties pane, use the light-bulb icon to change the name into “services.<type><area>.<servicename>.Server-BW01” (only the package name should change).

2. Do the same for the JNDIClient-ESB01 resource.

3. Do the same for the JMSClient-ESB01 resource.

4.2.4 Create the Process Packages

1. Create the process package “services.<type>.<area>.<servicename>”.

2. Move the BSService process into it.

3. Create the process package “services.<type>.<area>.<servicename>.sub”.

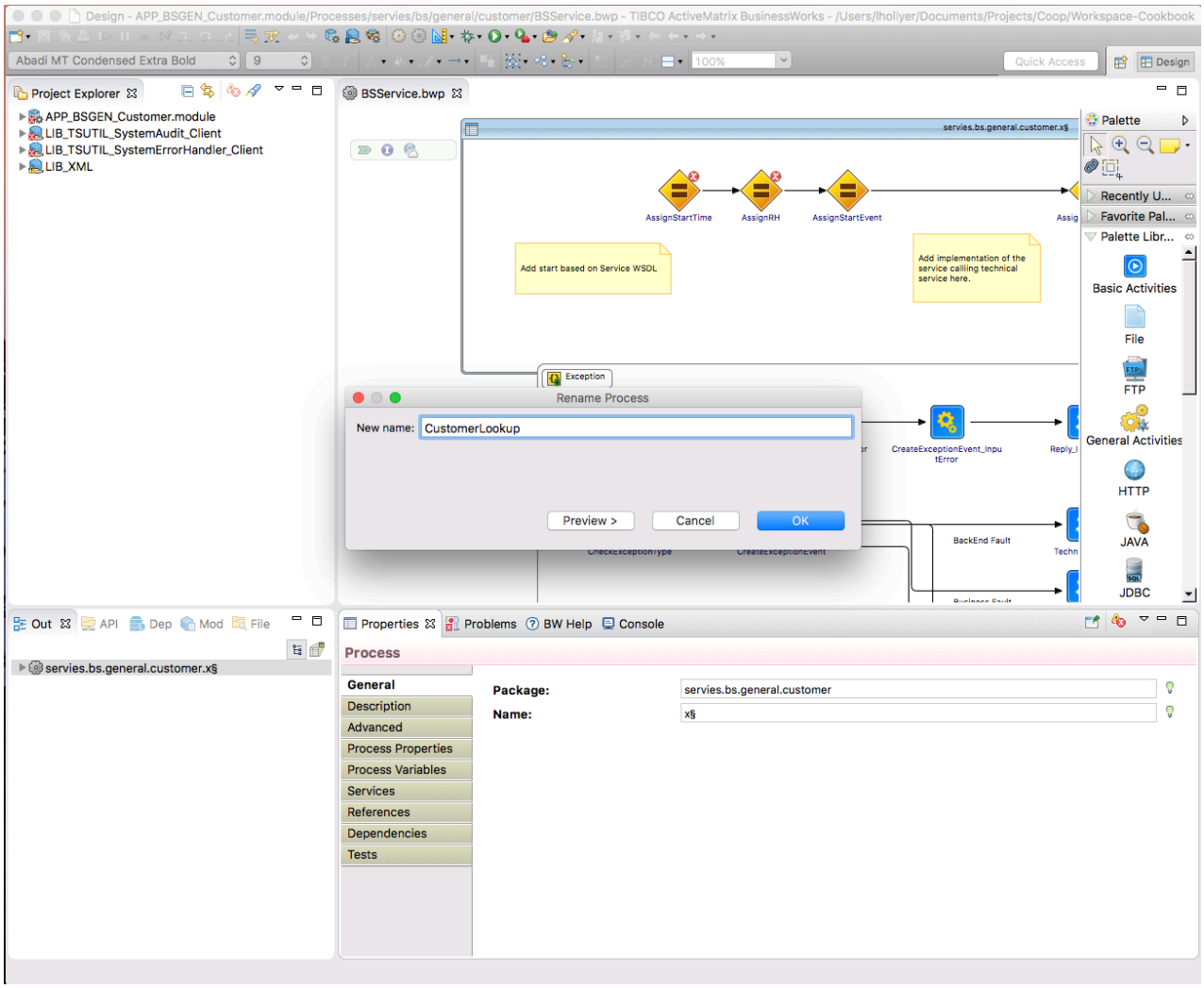
4. Move the Activator process into it.

5. Delete the remaining …ServiceName… process packages.

4.2.5 Change the Service Process

1. Open the process “services.<type>.<area>.<servicename> / BSService”.

2. In the properties, use the light-bulb icon to rename it into “<ServiceName>”:



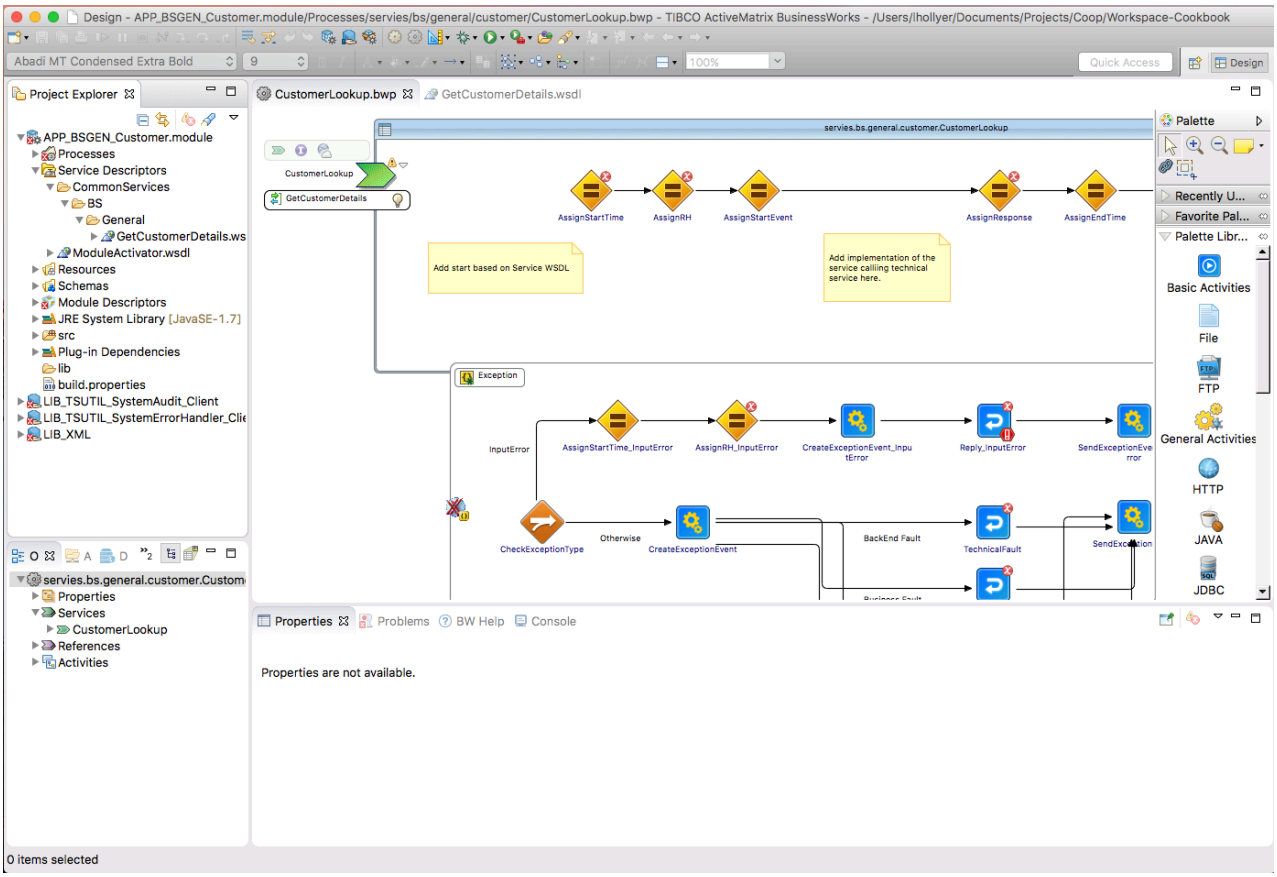
3. Import the XSD and WSDL into the module.

   a. Follow the **ServiceName.xsd** example in **Schemas / CommonServices / … / ServiceName** and make sure the targetNamespace and tns match up in the file directory, and the simple and complex types are renamed.

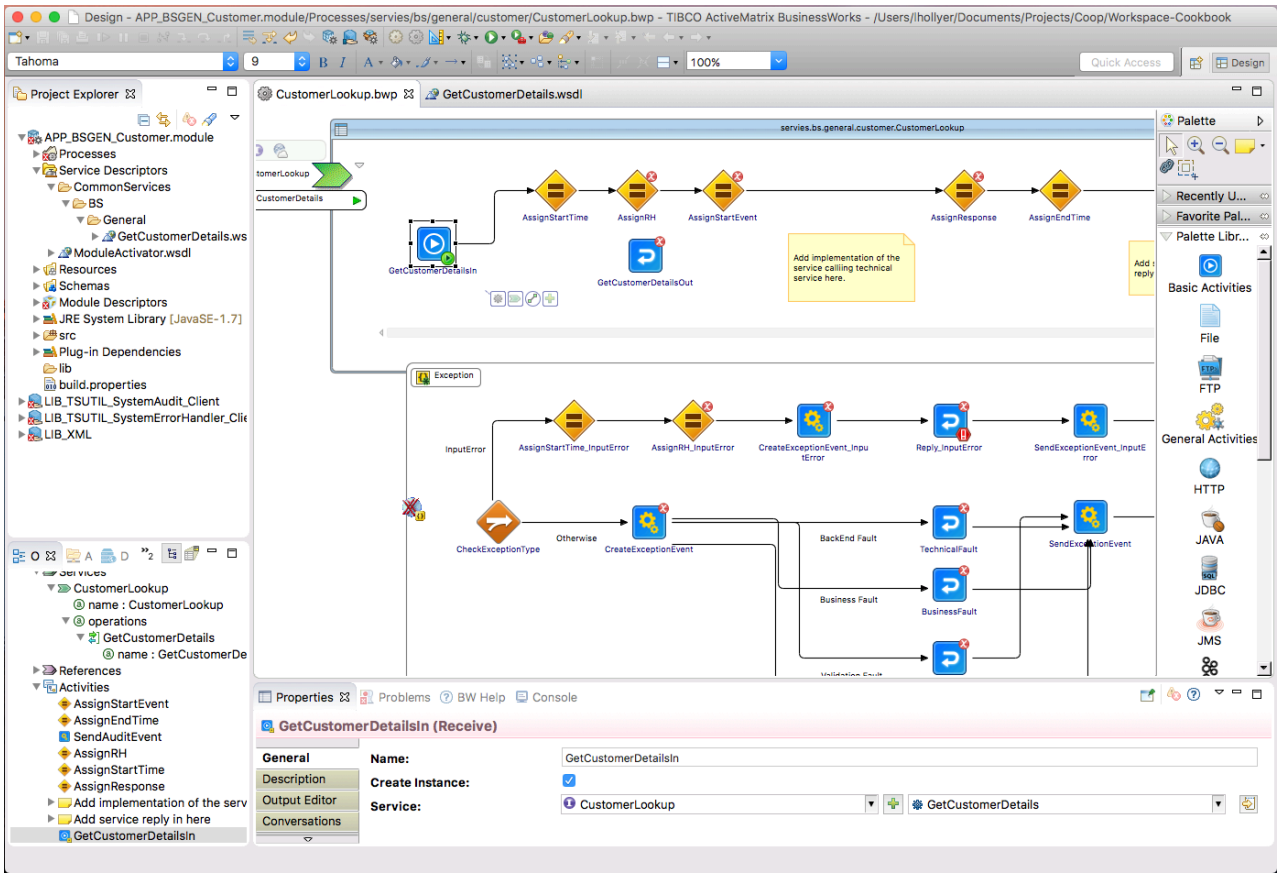
   b. Ensure **ServiceName.wsdl** located in **Service Descriptors / CommonServices / … / ServiceName** has message names and inputs/outputs matching up with **ServiceName.xsd**.

      In case they do not match up, change the imports in the source code in **ServiceName.wsdl** to match the file structure.

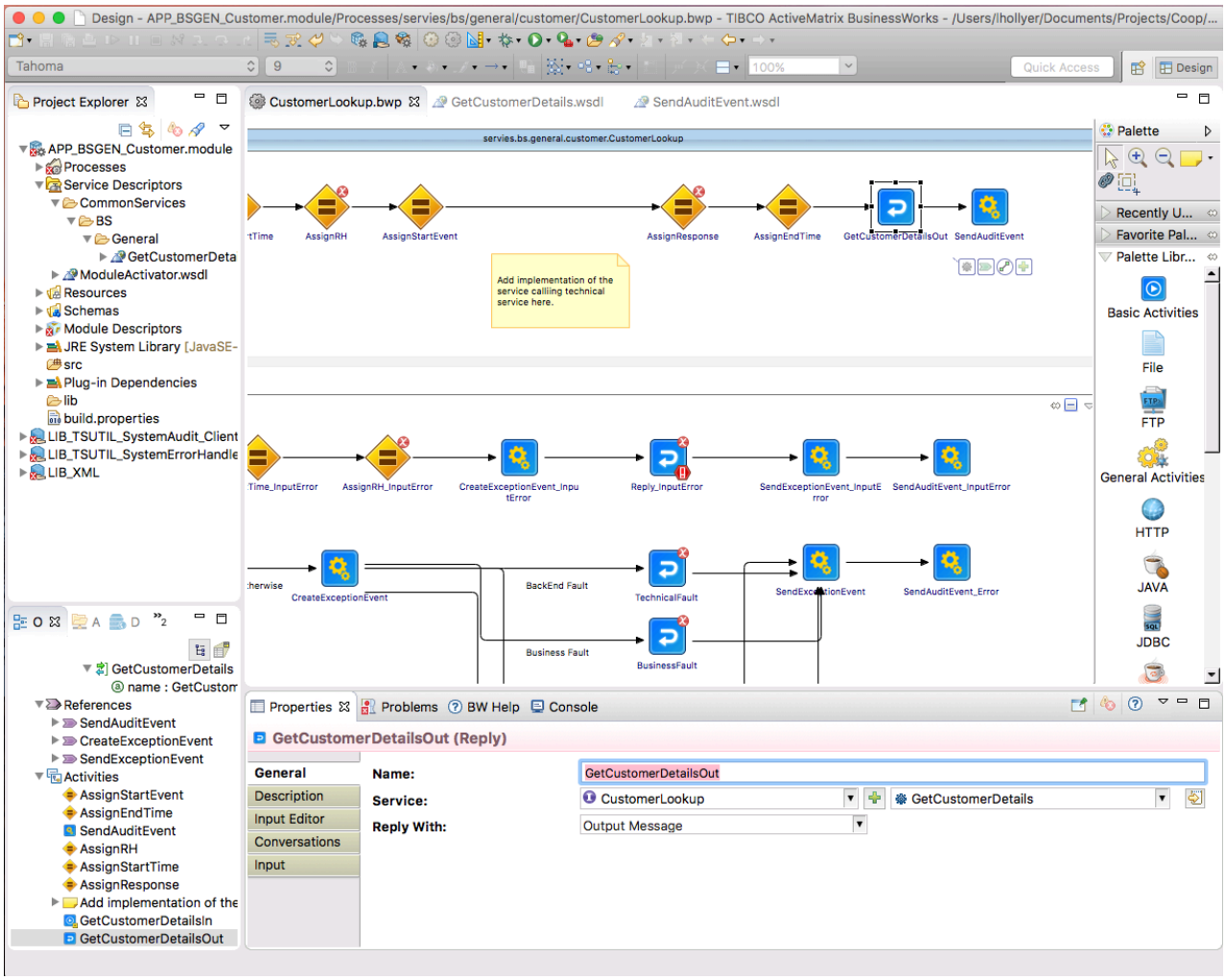
4. Drag and drop the abstract WSDL into the process so the service operation is implemented:



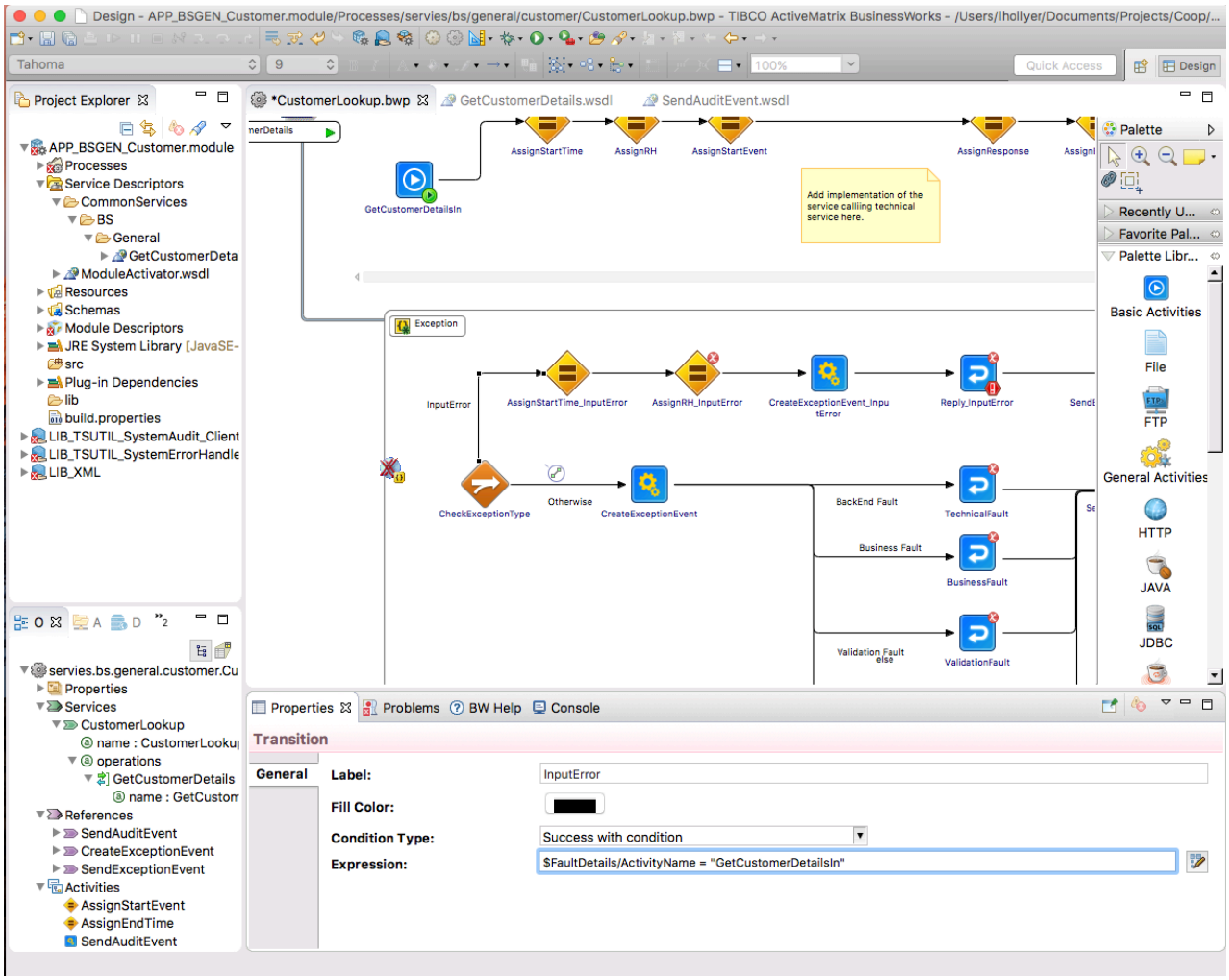
5. Create a transition between the <Operation>In task and “AssignStartTime” task:



6. Re-arrange transitions so the <Operation>Out task is located between “AssignEndTime” and “SendAuditEvent” tasks:

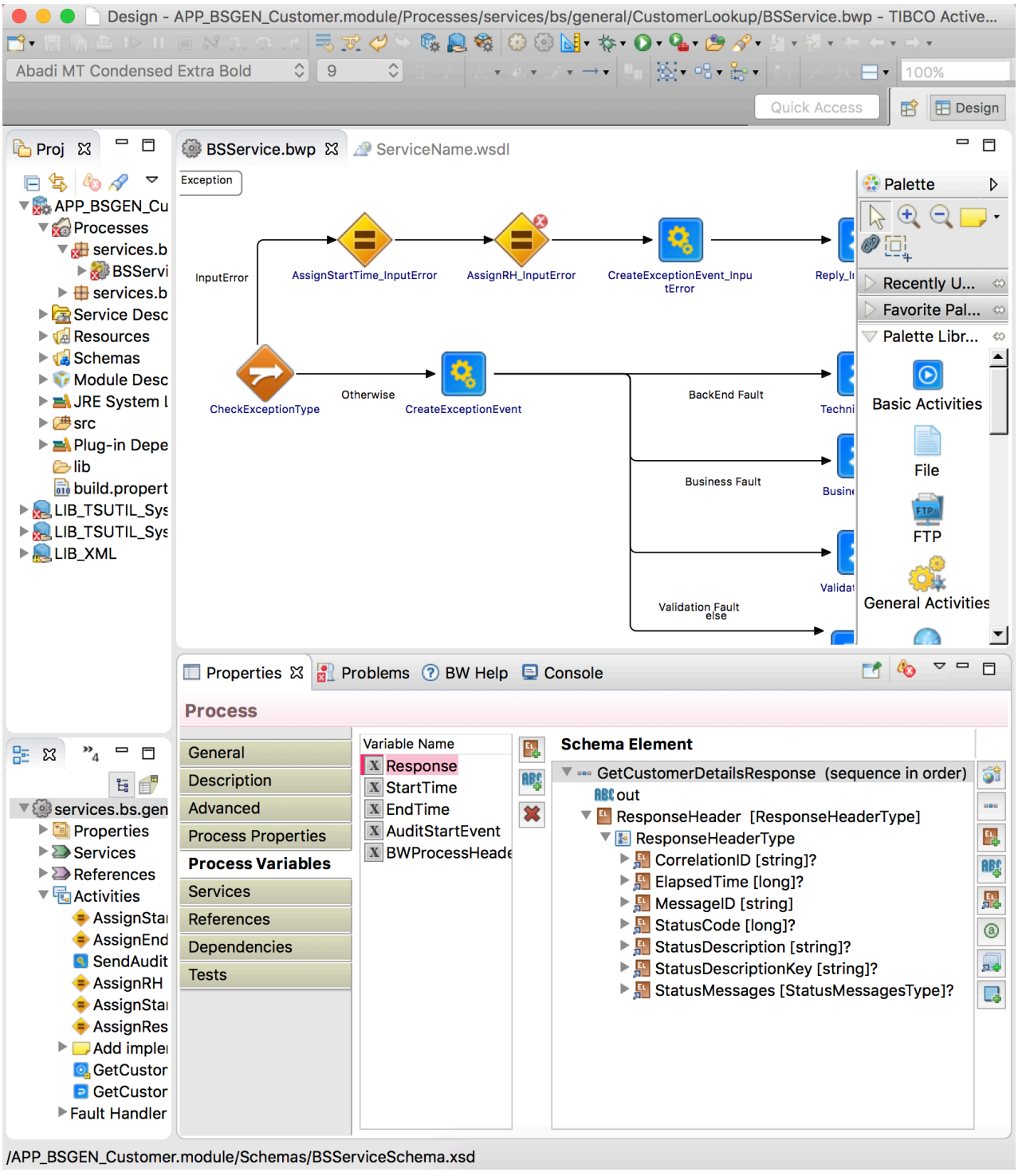


7. Fix the condition in the transition named “InputError” in the catch-all block as following: $FaultDetails/ActivityName = "<Operation>In".



8. Go the Process Variables. Change the schema of the “Response” variable so it uses the “<Operation>Response” element from the service XSD:

   If the response header doesn’t change automatically, add a new element called “ResponseHeader” and select the schema to be “ResponseHeaderType”. The child elements should automatically be added.



9. Save your changes.

4.2.6 Fix the Process Mappings

4.2.6.1 AssignRH

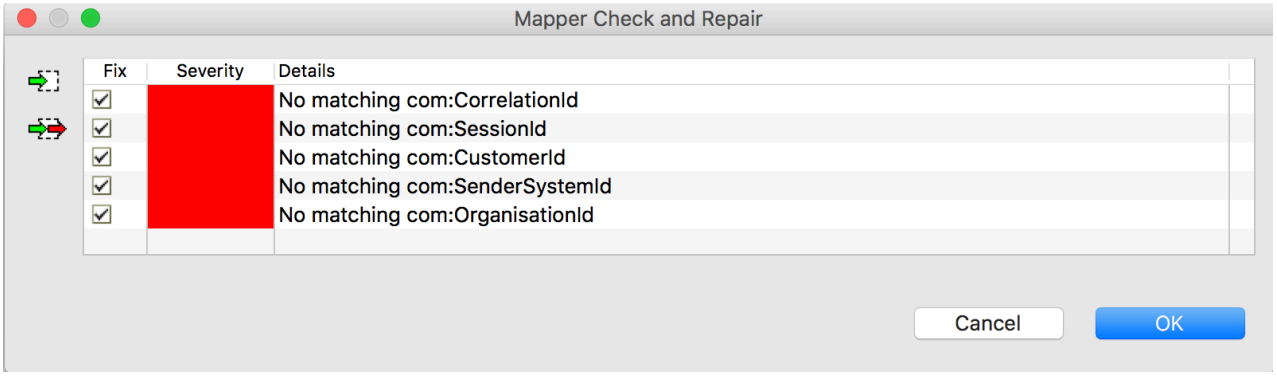
This task prepares the message header that can be passed into calls to other ESB services or returned in the reply or fault messages.

Fix the mapping in the task “AssignRH” as following:

1. Change the formula for the “varRequest” variable into

```
$<Operation>In/parameters/tns9:<Operation>Request
```

2. Fix the mapping of the CorrelationId in the When statement, if necessary (use the fixing icon from Eclipse):

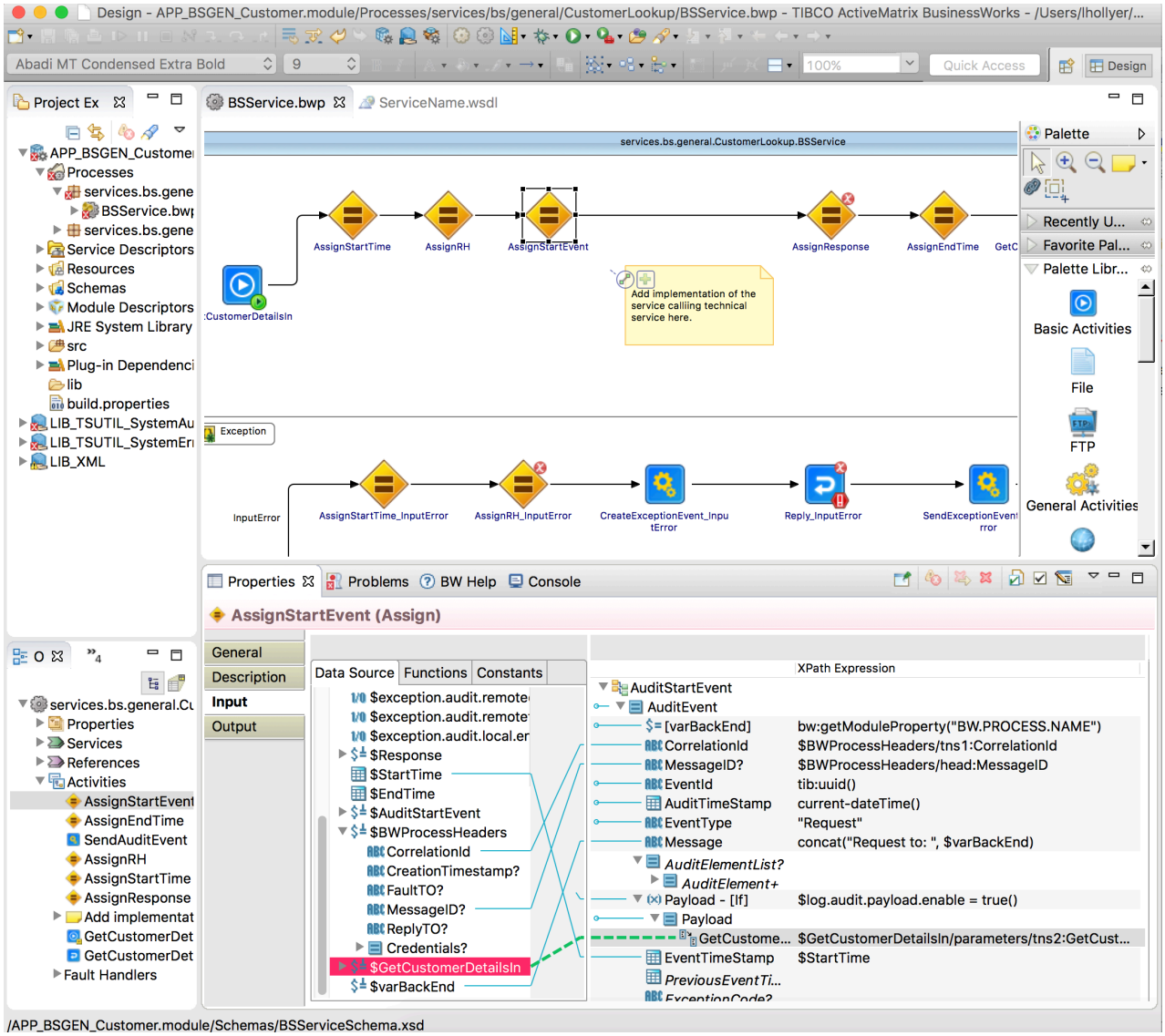


4.2.6.2 *AssignStartEvent*

This task prepares the Audit Event indicating this service has receive a request from a consumer. This event will be passed into the SendAuditEvent and SendAuditEvent_Error tasks.

Fix the mapping in “AssignStartEvent” as following:

1. Payload content should be a copy of the <Operation>In/parameters/<Operation>Request XML element. The “Payload” element must have a surrounding “If” statement which formula is “$log.audit.payload.enable = true()”:



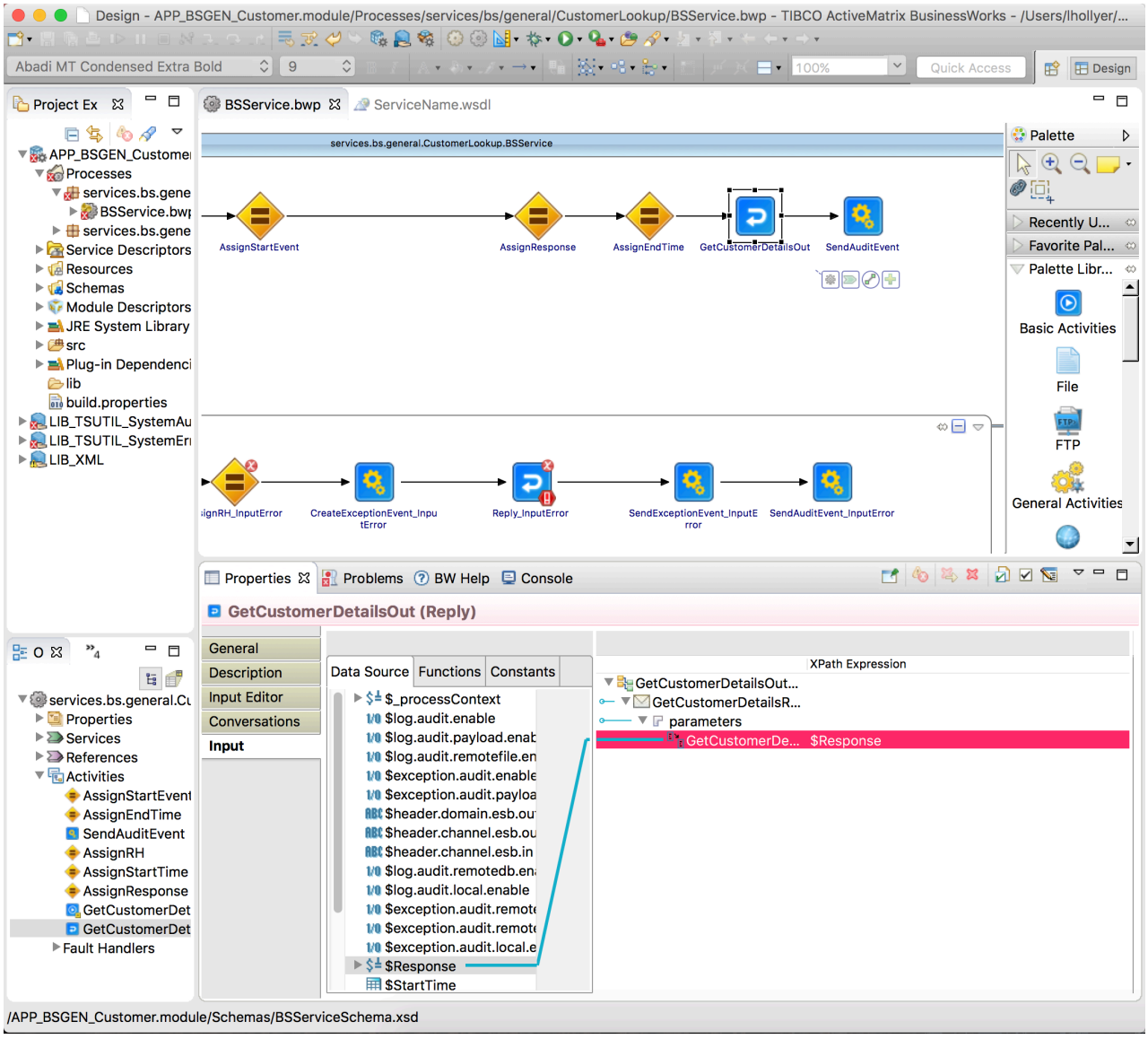
2.

4.2.6.3 *<Operation>Out*

This task returns a response to the consumer in case of success.

Fix the mapping as following:

1. AssignResponse – map “out” and “ResponseHeader from $Response.
2. <Operation>Response/parameters/<Operation>Response should be a copy-of the $Response XML element:



4.2.6.4 *Reply_InputError*

This tasks returns a ValidationFault SOAP fault to the consumer.

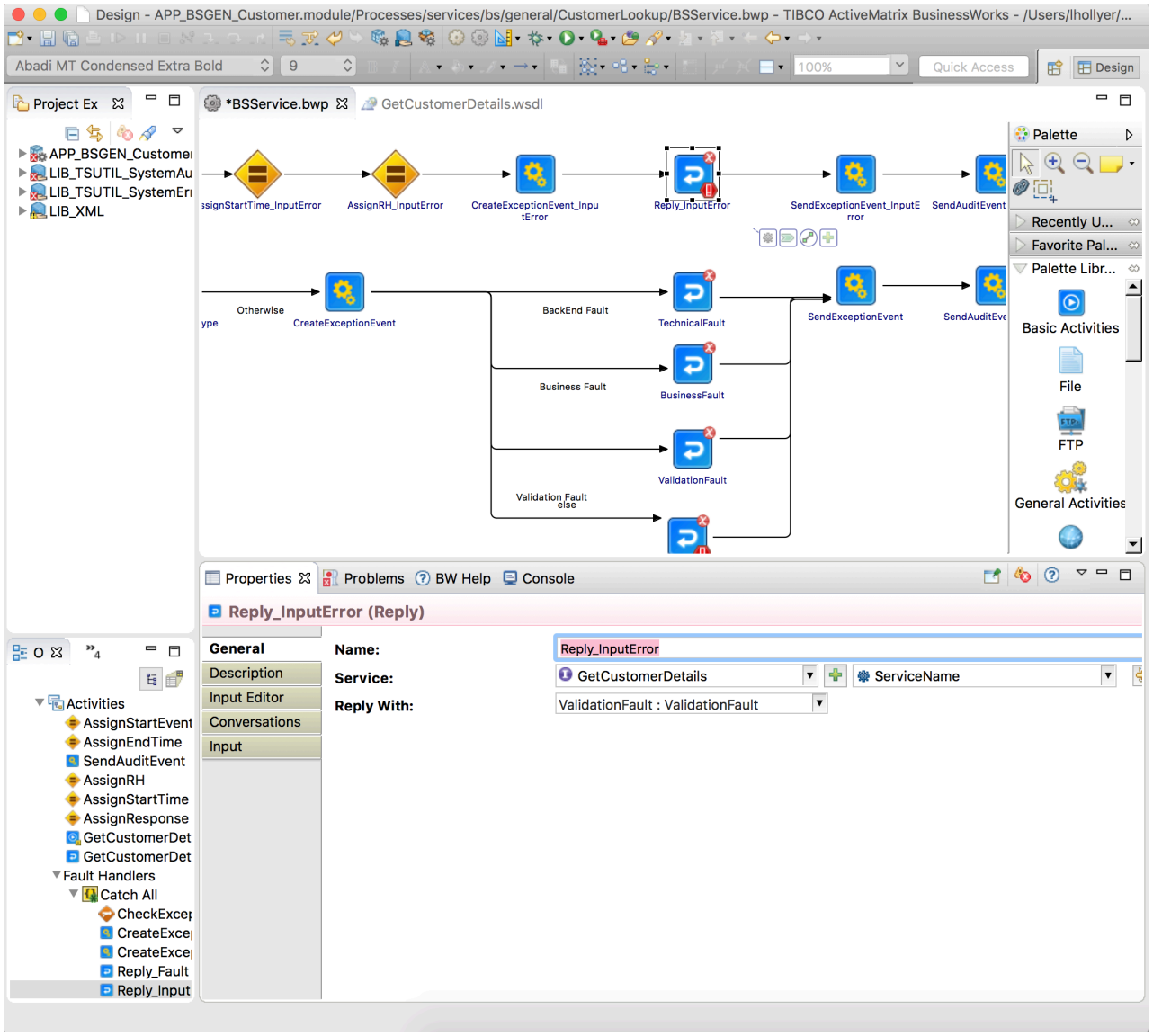
1. Change AssignRH_InputError/varRequest to:
   $GetCustomerDetailsIn/parameters/tns1:GetCustomerDetailsRequest

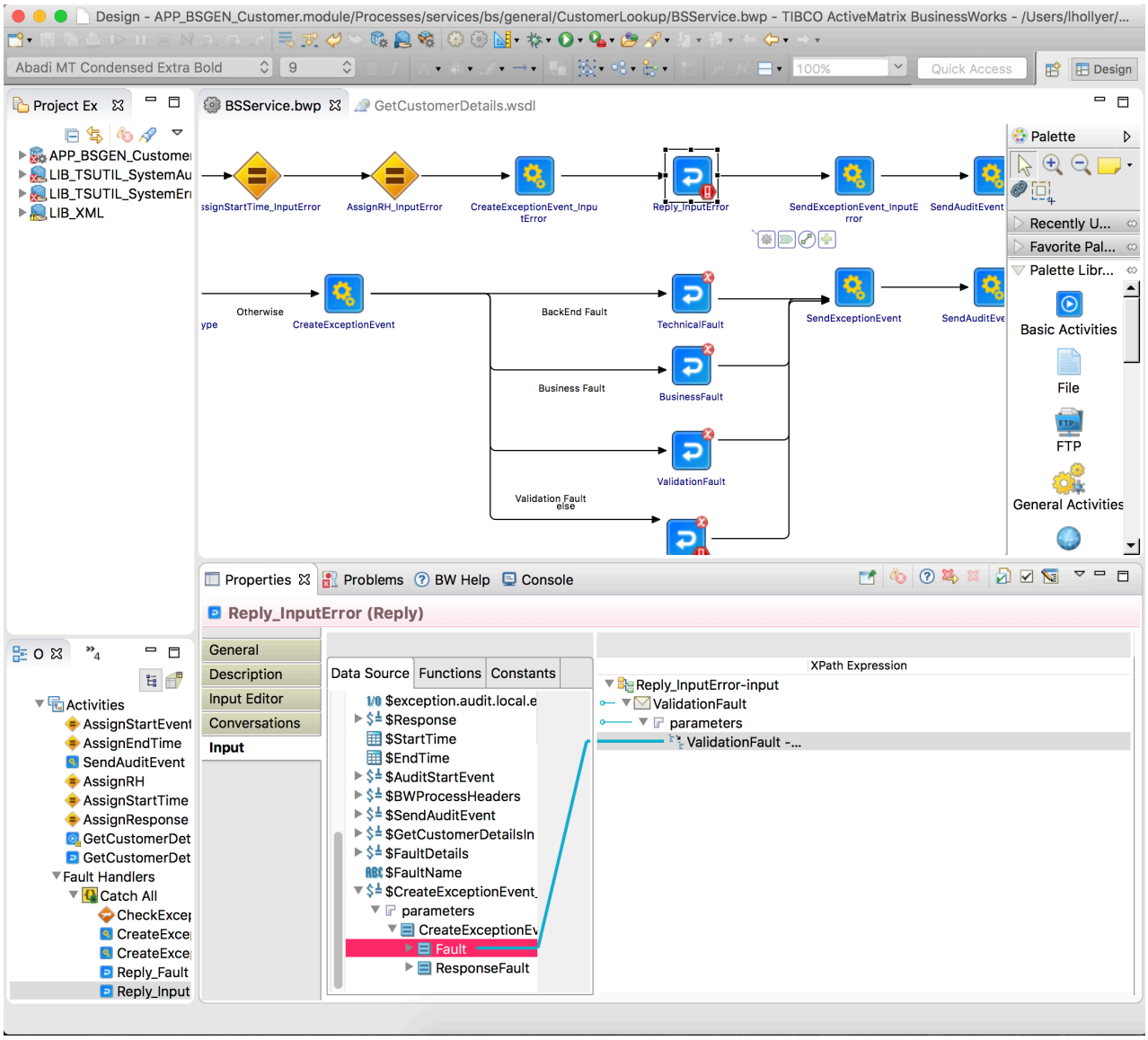
Fix the configuration of the Reply_InputError task (select the service, operation and “ValidationFault”). **If a concrete WSDL is used, change ‘Reply With’ to ‘Undeclared Fault’ instead.**

1. Fix the mapping of the Reply_InputError task so ValidationFault element is a copy of the CreateExceptionEvent_InputError / Response / ValidationFault element.

   If ValidationFault element doesn’t appear, check on the General tab the ‘Service’ is <ServiceName> and ‘Reply With’ is set to ‘ValidationFault’

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.




Design - APP_BSGEN_Customer.module/Processes/services/bs/general/CustomerLookup/BSService.bwp - TIBCO ActiveMatrix BusinessWorks - /Users/lhollyer/...
Project Ex
BSService.bwp
GetCustomerDetails.wsdl
APP_BSGEN_Custome
LIB_TSUTIL_SystemAu
LIB_TSUTIL_SystemEr
LIB_XML
AssignRH_InputError
CreateExceptionEvent_Inpu tError
Reply_InputError
SendExceptionEvent_InputE rror
SendAuditEvent
Otherwise
CreateExceptionEvent
BackEnd Fault
TechnicalFault
SendExceptionEvent
Business Fault
BusinessFault
ValidationFault
Validation Fault else
Palette
Recently U...
Favorite Pal...
Palette Libr...
Basic Activities
File
FTP
General Activities
Properties
Problems
BW Help
Console
Reply_InputError (Reply)
General
Description
Input Editor
Conversations
Input
Data Source
Functions
Constants
$exception.audit.local.e
$Response
$StartTime
$EndTime
$AuditStartEvent
$BWProcessHeaders
$SendAuditEvent
$GetCustomerDetailsIn
$FaultDetails
$FaultName
$CreateExceptionEvent_
parameters
CreateExceptionEv
Fault
ResponseFault
XPath Expression
Reply_InputError-input
ValidationFault
parameters
ValidationFault -...
Activities
AssignStartEvent
AssignEndTime
SendAuditEvent
AssignRH
AssignStartTime
AssignResponse
GetCustomerDet
GetCustomerDet
Fault Handlers
Catch All
CheckExcep
CreateExce
CreateExce
Reply_Fault
Reply_Input

4.2.6.5 Reply_Fault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML Fault element.

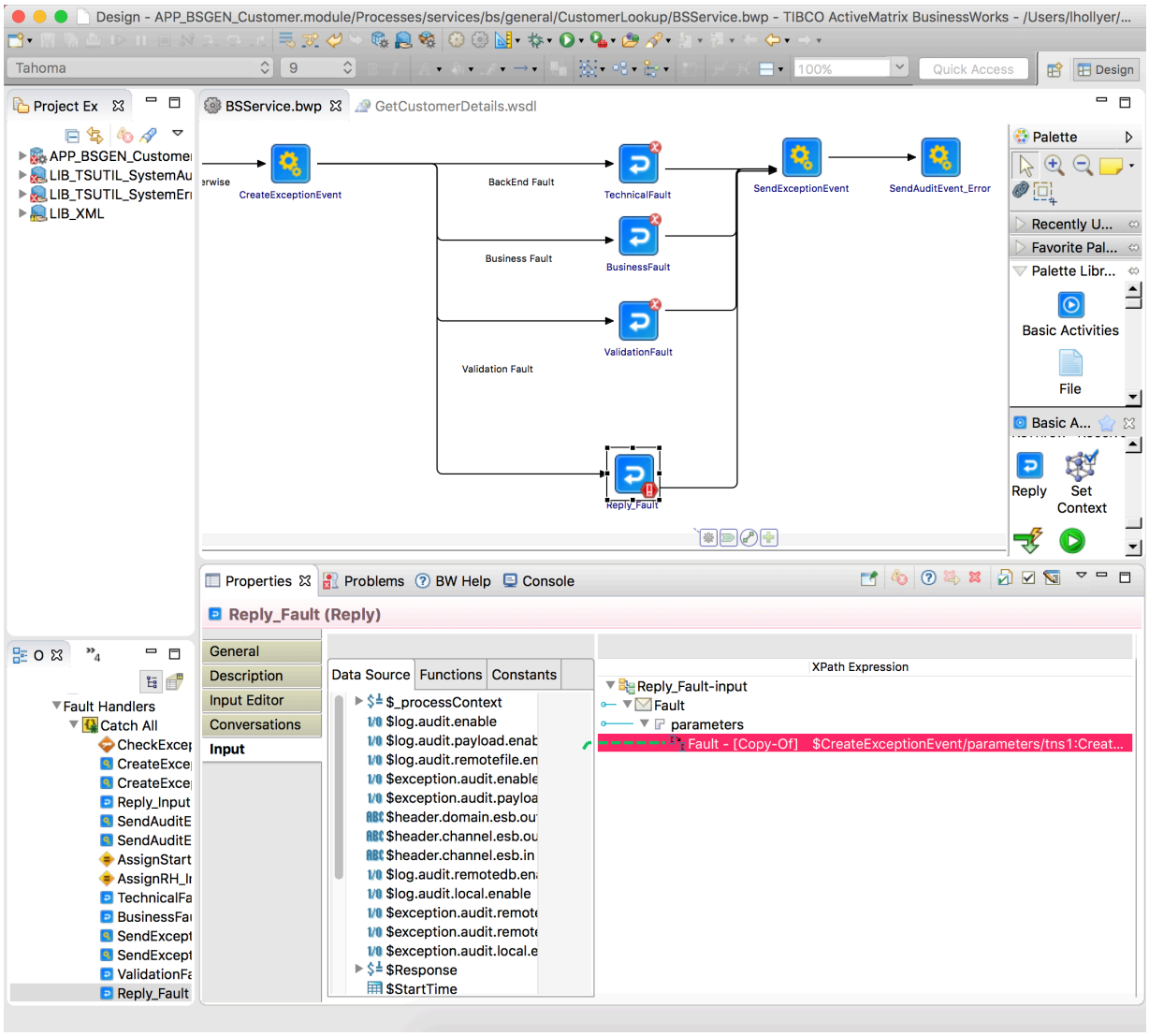
Fix the configuration of the Reply_Fault task:

1. Select the service, operation and “Fault”. **If a concrete WSDL is used, change ‘Reply With’ to ‘Undeclared Fault’ instead.**

Fix the mapping of the Reply_Fault task so.

Fault element is a copy of the CreateExceptionEvent / Response / Fault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.


Design - APP_BSGEN_Customer.module/Processes/services/bs/general/CustomerLookup/BSService.bwp - TIBCO ActiveMatrix BusinessWorks - /Users/lhollyer/...
BSService.bwp
GetCustomerDetails.wsdl
CreateExceptionEvent
BackEnd Fault
TechnicalFault
SendExceptionEvent
SendAuditEvent_Error
Business Fault
BusinessFault
Validation Fault
ValidationFault
Reply_Fault
Palette
Basic Activities
File
Reply
Set Context
Properties
Problems
BW Help
Console
Reply_Fault (Reply)
General
Description
Input Editor
Conversations
Input
Data Source
Functions
Constants
XPath Expression
Reply_Fault-input
Fault
parameters
Fault - [Copy-Of] $CreateExceptionEvent/parameters/tns1:Creat...
Fault Handlers
Catch All

4.2.6.6 TechnicalFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML TechnicalFault element.

Fix the configuration of the TechnicalFault task:

1. Select the service, operation and “TechnicalFault”. **If a concrete WSDL is used, change ‘Reply With’ to ‘Undeclared Fault’ instead.**

Fix the mapping of the TechnicalFault task so:

2. TechnicalFault element is a copy of the CreateExceptionEvent / Response / TechnicalFault element.

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

4.2.6.7 BusinessFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML BusinessFault element.

Fix the configuration of the BusinessFault task:

1. Select the service, operation and “BackEndFault”. **If a concrete WSDL is used, change ‘Reply With’ to ‘Undeclared Fault’ instead.**

Fix the mapping of the BusinessFault task so:

2. BusinessFault element is a copy of the CreateExceptionEvent / Response / BusinessFault element.

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

4.2.6.8 ValidationFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML ValidationFault element.

Fix the configuration of the ValidationFault task:

1. Select the service, operation and “ValidationFault”. **If a concrete WSDL is used, change ‘Reply With’ to ‘Undeclared Fault’ instead.**

Fix the mapping of the ValidationFault task so:

2. ValidationFault element is a copy of the CreateExceptionEvent / Response / ValidationFault element.

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent's condition is "Success with no matching condition".

4.2.7 Create the SOAP Binding

1. Create the SOAP Binding by clicking on "Create Service"
2. Drop and drag the newly created operation into the process.

4.3 Implement the Service

4.3.1 Business Logic

1. Implement the Service: put the logic task between the "AssignStartEvent" and "AssignResponse" tasks.

4.3.2 AssignResponse

1. The task AssignResponse maps the business logic data into the reply to be sent by the service.

You MUST map the ResponseHeader element as following:

a. CorrelationId: $BWProcessHeaders / CorrelationId

b. MessageId: $BWProcessHeaders / MessageId

c. Version: $BWProcessHeaders / Response / Version

d. ResponseDateTime: current-dateTime()

4.3.3 Error Handling

1. Depending on the design, some services may return a valid response in case of fault. In such case:

a. edit the mapping AssignErrorResponse task

b. Change the configuration of the Reply_Fault task so it return a response and the mapping use the $Response process variable.

c. You can do the same for other branches of fault if certain faults must return a valid reply and not a fault:

i. Add an Assign task to assign values into the $Response variable.

ii. Change the configuration and mapping of the XXXFault task so it return the $Response as valid response instead of a fault.

4.3.4 Invoke another ESB service

1. If you invoke another ESB service (BS or TS), you MUST map the RequestHeader element as following:

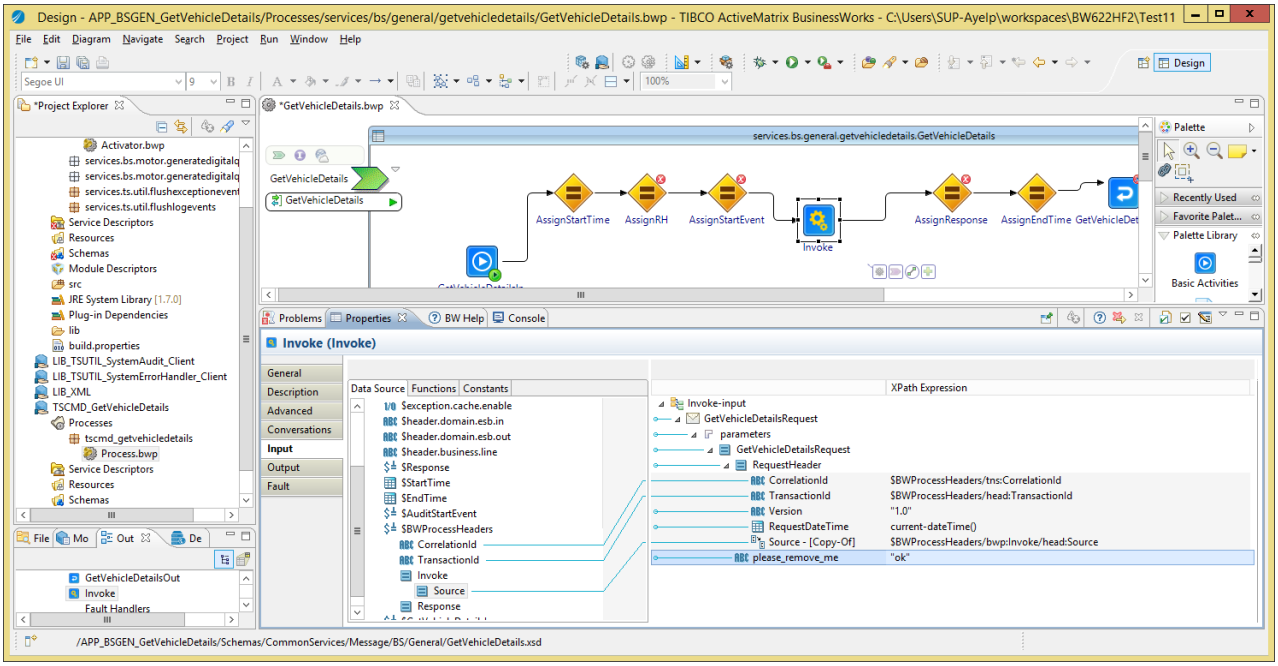
    a. CorrelationId: $BWProcessHeaders / CorrelationId

    b. MessageId: $BWProcessHeaders / MessageId

    c. Version: The version of the invoked service, for example “1.0”.

    d. RequestDateTime: current-dateTime()

In this screenshot we invoke TSGEN_GetVehicleDetails TS service:



4.3.5 Validate the Request

Some service may require additional input data validation (besides what is defined inside the service schema, which BW validates by default). In such case, you must implement the validation as following:

For a single validation rule:

1. Add a Throw task to the process,
2. Rename the task into “Throw_ValidationFault”
3. Create a transition from the “AssignStartEvent” task to the “Throw_ValidationFault” task.
4. Change the transition condition type to “Success with Condition”.
5. In the “Expression” field, type the validation condition which will evaluate to “true” if the rule is broken.
6. Change the condition type of the transition between “AssignStartEvent” and “AssignStartEvent” into “Success with no matching condition”:

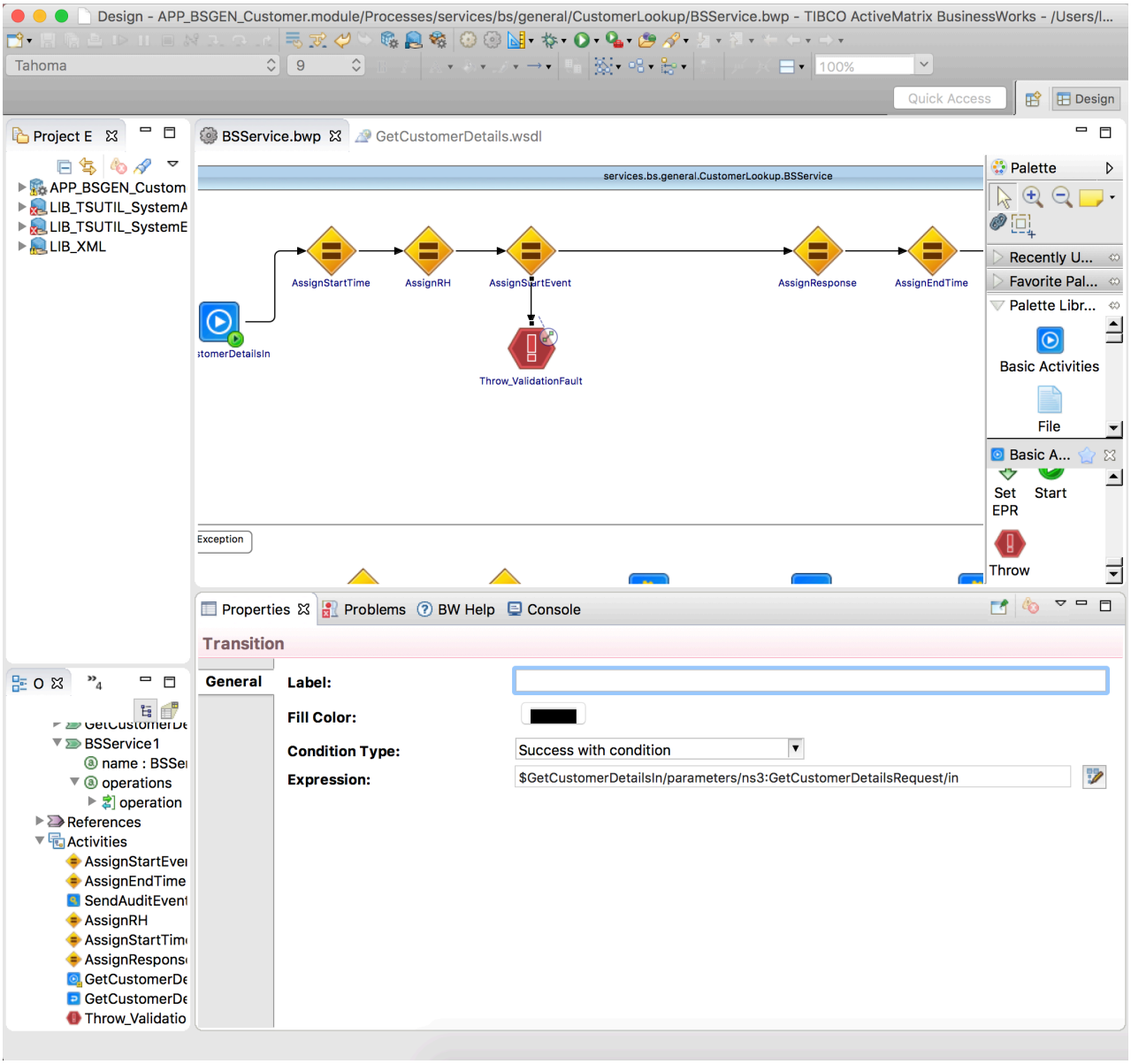


Figure 1: Throw_ValidationFault sample

7. Change the task “Throw_ValidationFault” as following:

    1. In “Input Editor”, add the XML element “ThrowableValidationFault” from the “LIB_XML / Schemas / CommonServices / Data / Internal / 1.0 / Throwable” XSD.

    2. Change the input mapping as following:

    3. ExceptionTimestamp: current-dateTime()

    4. ExceptionDescription: a description of the issue, for example concat(“Customer is incorrect: “, $xxx). When you mention an input element value is incorrect, always put its value in the error description (e.g. $xxx).

For multiple rules:

- The steps are the same, but you insert a mapping task that will make multiple verifications and output a list or error descriptions. The “Throw_ValidationFault” task will concatenate all of them into a single ExceptionDescription.

4.4 Finalization

If the service does not invoke any other service via JMS, you must:

1. Delete the resource JNDIClient-ESB01.
2. Delete the resource JNDIClient-ESB01.
3. Delete the property group **resources / services.bs.<area>.<servicename> / Client-ESB01**.

Notes:

- TSUTIL_SystemAudit_Client and TSUTIL_SystemErrorHandling_Client use their own JMS connection resources so deleting the resources from the Business service is not an issue.

5 Create a Business Service Provider as a Shared Module

This chapter explains how to create a new Business service as a Shared Module, only accessible as process-call from another BW Business service.

5.1 Template Overview

The template module APP_BSServiceTemplate.module allows you to create a new application for a Business service. It contains the following objects:

- Processes:
  - **BSService**: the service implementation.
- Resources:
  - **JNDIClient-ESB01**: the JNDI connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.
  - **JMSClient-ESB01**: the JMS connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.

Notes:

2 The JNDI and JMS connections used by the System Error Handler and System Audit are separated from JNDIClient-ESB01 and JMSClient-ESB01. They are contained in the LIB_TSUTIL_SystemErrorHandler_Client and LIB_TSUTIL_SystemAudit_Client modules.

5.2 Pre-Requisites

The XSD and WSDL have been created.

5.3 Procedure

5.3.1 Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “LIB_BSServiceTemplate.module” into the folder **<GIT> / trunk / BW / BusinessServices**.
2. Rename the copied folder into “LIB_BS<Area>_<ServiceName>”.
3. With a text editor, open the .project file into the copied folder and change the project name at the top from LIB_BSServiceTemplate.module into LIB_BS<Area>_<ServiceName>.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>LIB_BSGEN_ServiceA</name>
```

```
        <comment></comment>
…
```

With TIBCO BusinesStudio

4. Start TIBCO BusinessStudio
5. In your workspace, import the following modules:
    e. LIB_XML (from <GIT> / trunk / XML
    f. LIB_TSUTIL_SystemAudit_Client (from <GIT> / trunk / BW / TechnicalServices)
    g. LIB_TSUTIL_SystemErrorHandler_Client (from <GIT> / trunk / BW / TechnicalServices)
    h. ~LIB_BS<Area>_<ServiceName> (from <GIT> / trunk / BW / BusinessServices)

All subsequent changes are done on the APP_<TYPE><Area>_<ServiceName> module:

6. Go to the module overview.

7. Change the name into “LIB_BS<Area>_<ServiceName> Module”:

5.3.2 Module Properties, Processes, Service Invocation

The rest of the procedure is the same as for a business service provider in an Application, please refer to the previous chapter for all required procedure information, except for the next sections, which are specific to an shared module.

5.4 Finalization

If the service does not invoke any other service via JMS, you must:

4. Delete the resource JNDIClient-ESB01.

5. Delete the resource JNDIClient-ESB01.

6. Delete the property group **resources / services.bs.<area>.<servicename> / Client-ESB01**.

Notes:

- TSUTIL_SystemAudit_Client and TSUTIL_SystemErrorHandling_Client use their own JMS connection resources so deleting the resources from the Business service is not an issue.

6 Create a Technical Service Provider as a Shared Module

This chapter explains how to create a Technical Service as a Shared Module, not exposed on JMS nor HTTP, only accessible as process-call from another BW service.

6.1 Template Overview

The template module “APP_ServiceTemplate.module” (renamed as APP_TSServiceTemplate.module) allows you to create a new application for a Business service. It contains the following objects:

- Processes:
  - **TSService**: the service implementation.
- Resources:
  - **Client-BACKEND-http:** the HTTP client the Technical service MUST use if the back-end system is exposed on HTTP/HTTPS transport.
  - **Client-BACKEND-jdbc:** the JDBC client the Technical service MUST use if the back-end system is exposed as a database.

Notes:

3 The JNDI and JMS connections used by the System Error Handler and System Audit are contained in the LIB_TSUTIL_SystemErrorHandler_Client and LIB_TSUTIL_SystemAudit_Client modules.

6.2 Pre-Requisites

The XSD and WSDL have been created. **If you are using an existing concrete WSDL, see section 6.5.**

6.3 Procedure

6.3.1 Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “LIB_TSServiceTemplate.module” into the folder **<GIT> / trunk / BW / TechnicalServices**.
2. Rename the copied folder into “LIB_TS<Area>_<ServiceName>”.
3. With a text editor, open the .project file into the copied folder and change the project name at the top from LIB_TSServiceTemplate.module into LIB_TS<Area>_<ServiceName>.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>LIB_TSCMD_GetVehicleDetails</name>
	<comment></comment>
```

…

With TIBCO BusinesStudio

1. Start TIBCO BusinessStudio
2. In your workspace, import the following modules:
    a. LIB_XML (from <GIT> / trunk / XML
    b. LIB_TSUTIL_SystemAudit_Client (from <GIT> / trunk / BW / TechnicalServices)
    c. LIB_TSUTIL_SystemErrorHandler_Client (from <GIT> / trunk / BW / TechnicalServices)
    d. LIB_TS<Area>_<ServiceName> (from <GIT> / trunk / BW / TechnicalServices)

All subsequent changes are done on the LIB_<TYPE><Area>_<ServiceName> module:

3. Go to the module overview.

4. Change the name into “LIB_<TYPE><Area>_<ServiceName> Module”:

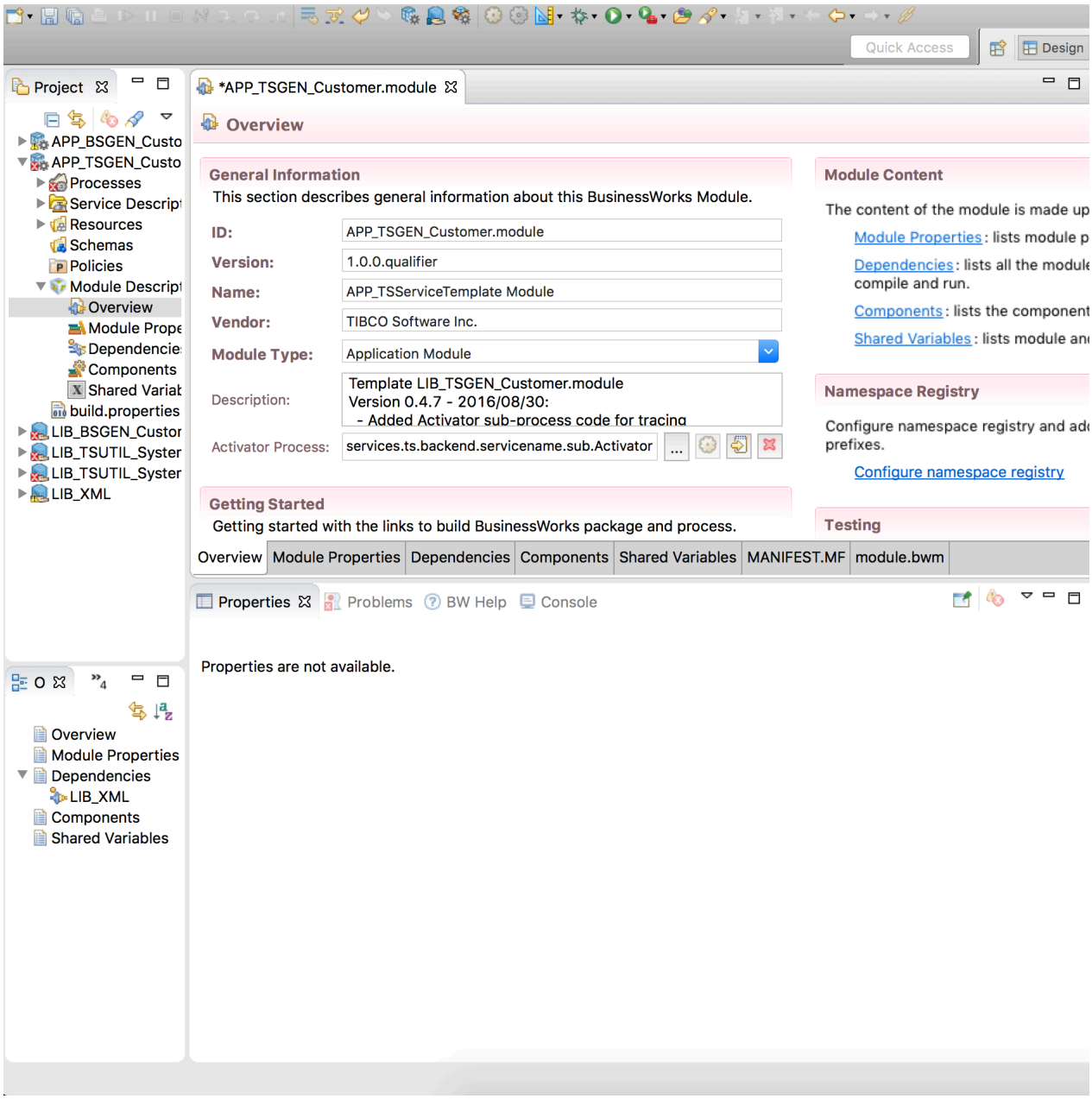


Figure 2: TS Service Provider Shared Module: Module Name

6.3.2 Change the Module Properties

1. Go to the module properties.

2. Select the group “services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”:

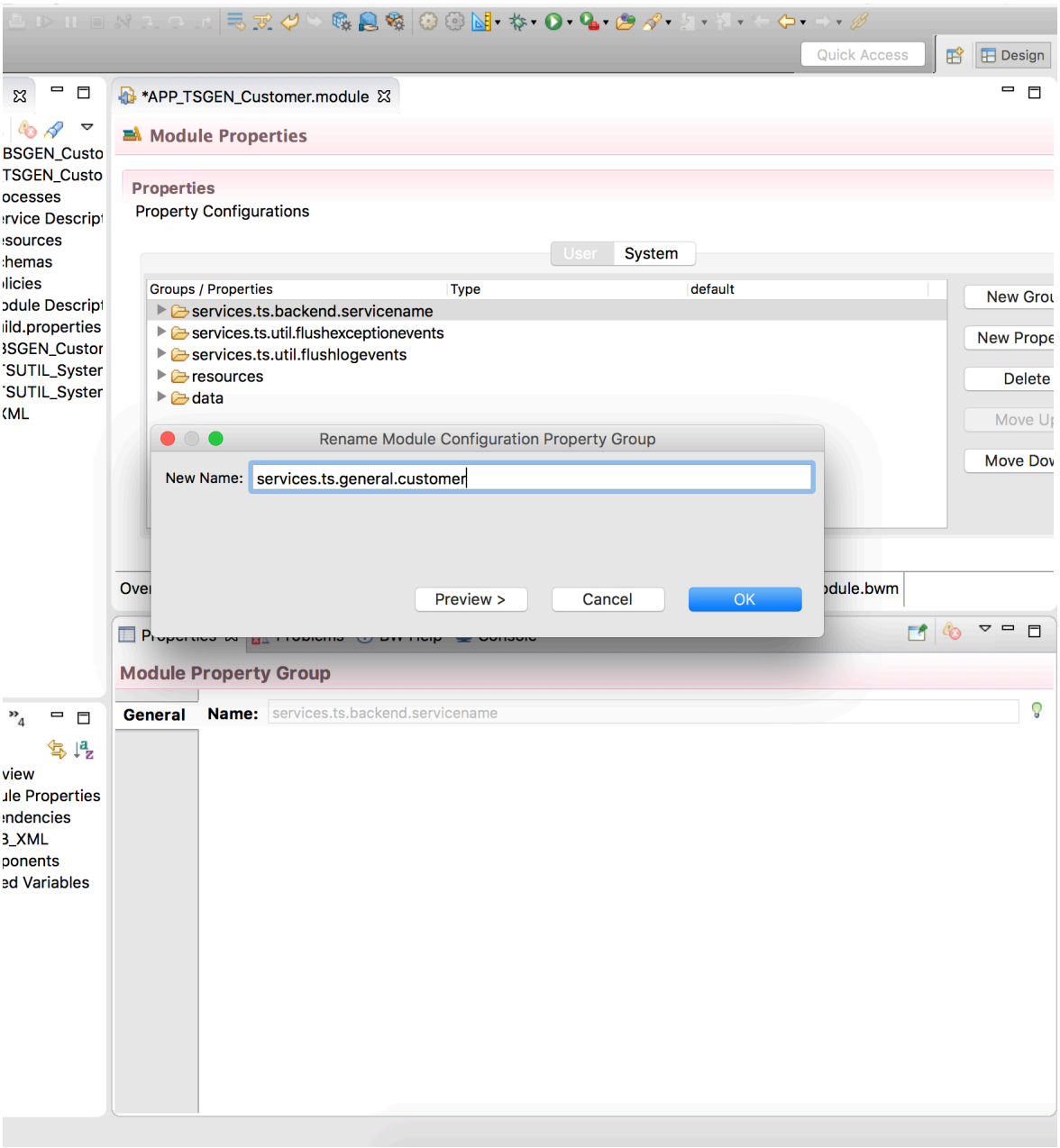
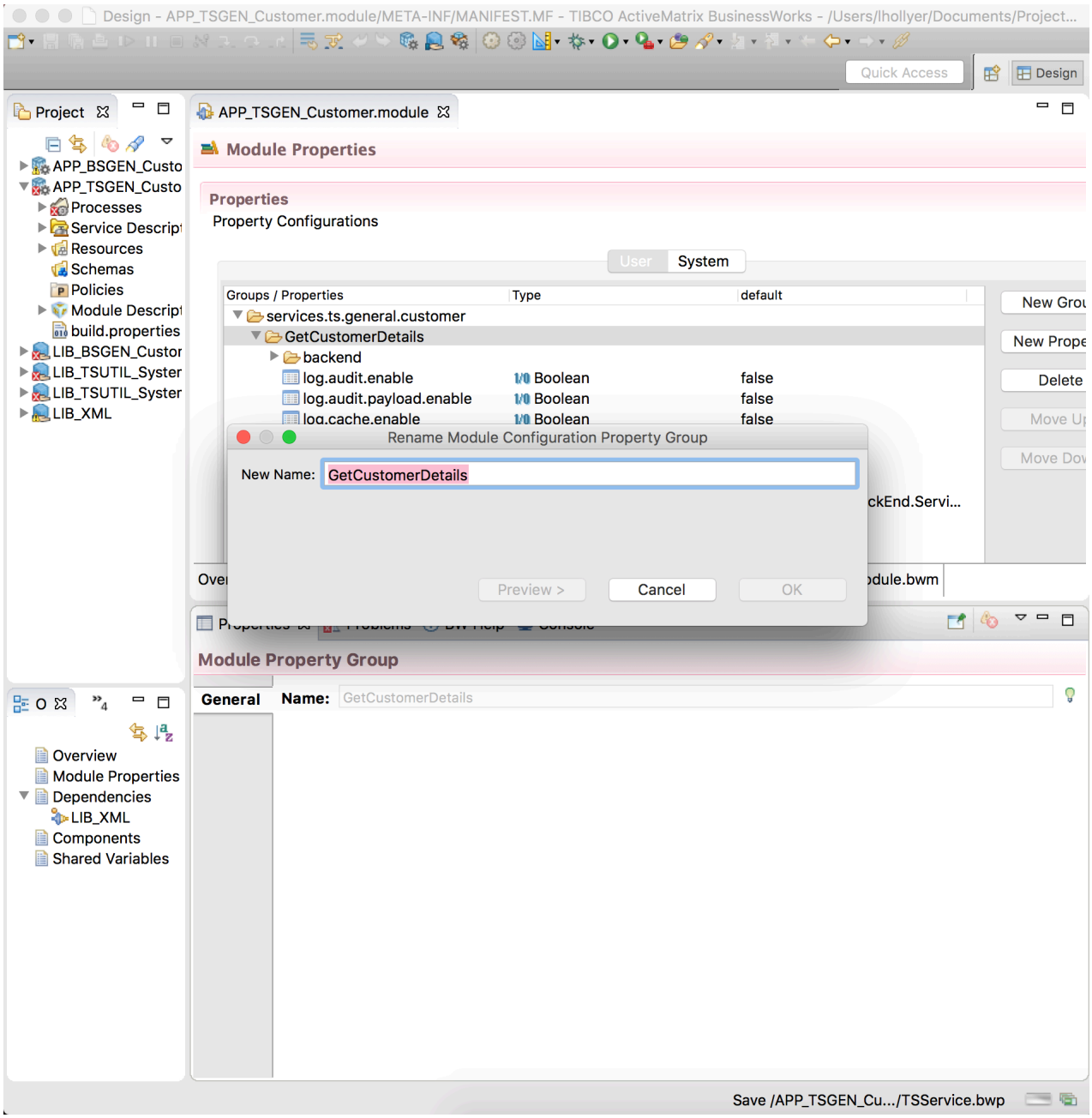


Figure 3: TS Service Provider Shared Module: Module Properties

3. In this group, use the same technique to rename the group “TSService” into “<ServiceName>” (for example “GetCustomerDetails”).



4. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”:

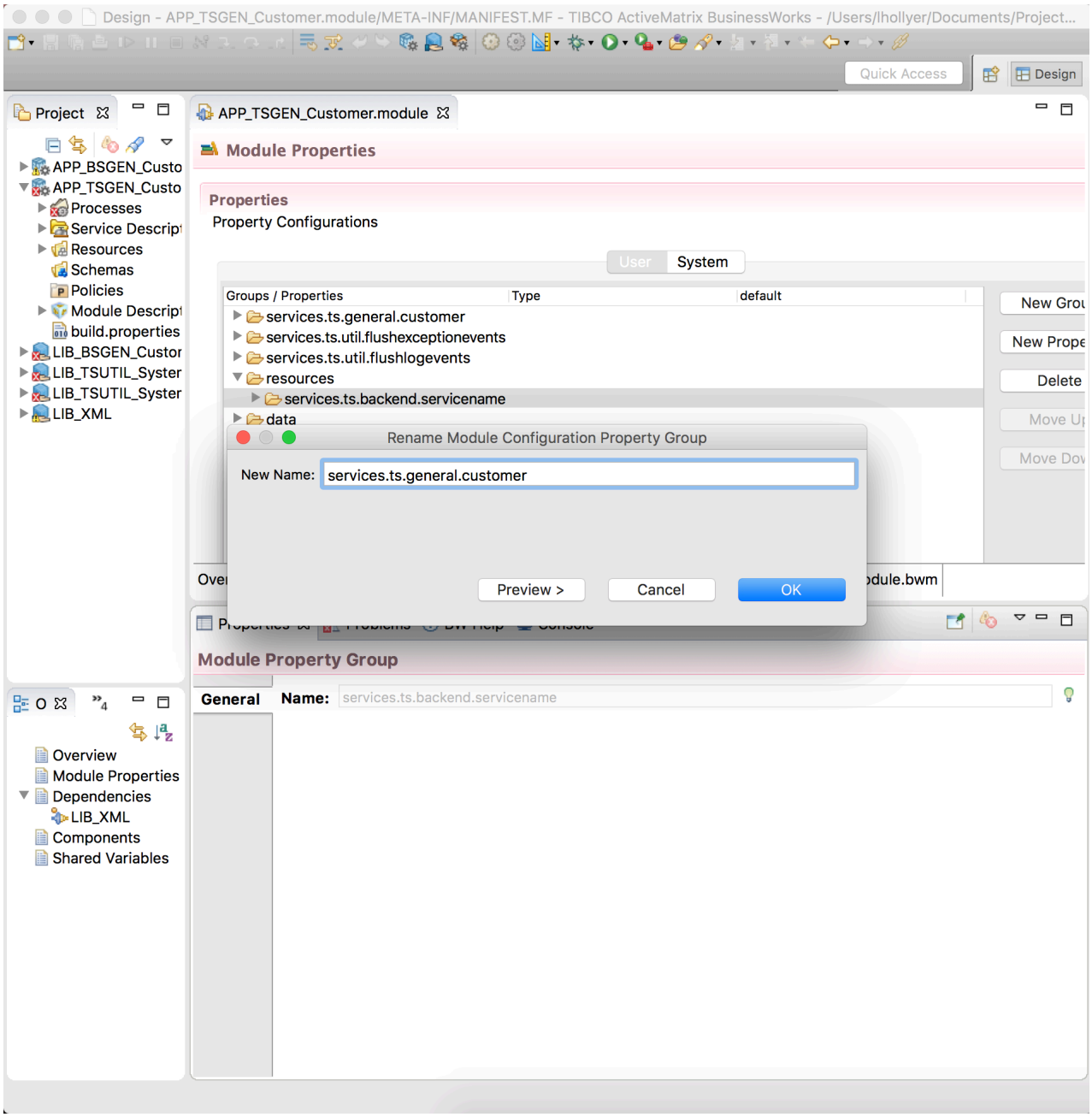


Figure 4: TS Service Provider Shared Module: Resource Properties

6.3.3 Create the Process Packages

1. Create the process package “services.ts.<area>.<servicename>”.
2. Move the TSService process into it.
3. Create the process package “services.<type>.<area>.<servicename>.sub”.
4. Move the Activator process into it.

5. Delete the remaining …servicename… process packages.

6.3.4 Change the Service Process

1. Open the process “services.ts.<area>.<servicename> / TSService”.

2. In the properties, use the light-bulb icon to rename it into “<ServiceName>”:

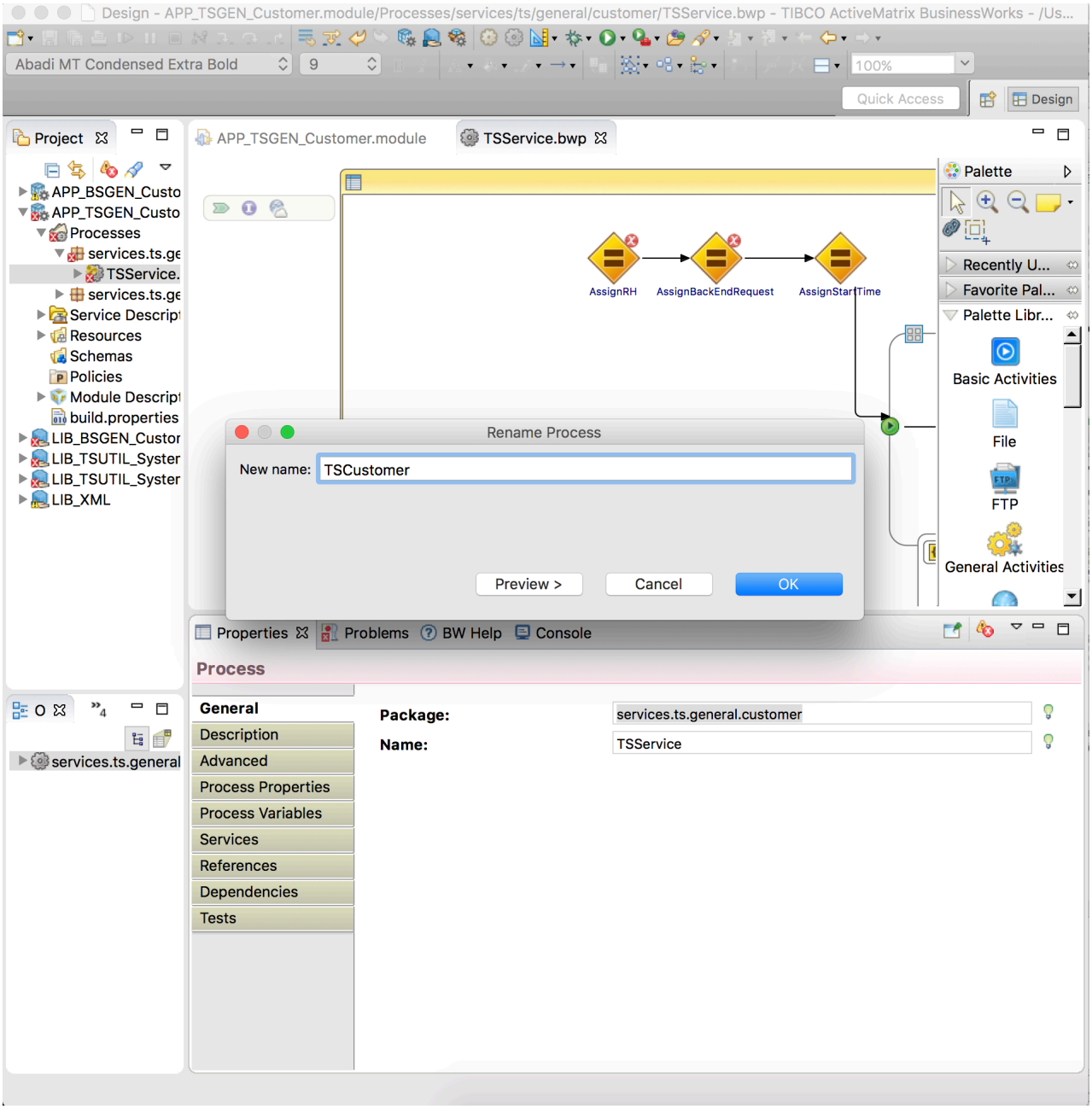


Figure 5: TS Service Provider Shared Module: Process Name

3. Import the XSD and service WSDL of the TS Service into the module.

4. Drag and drop the abstract WSDL into the process:

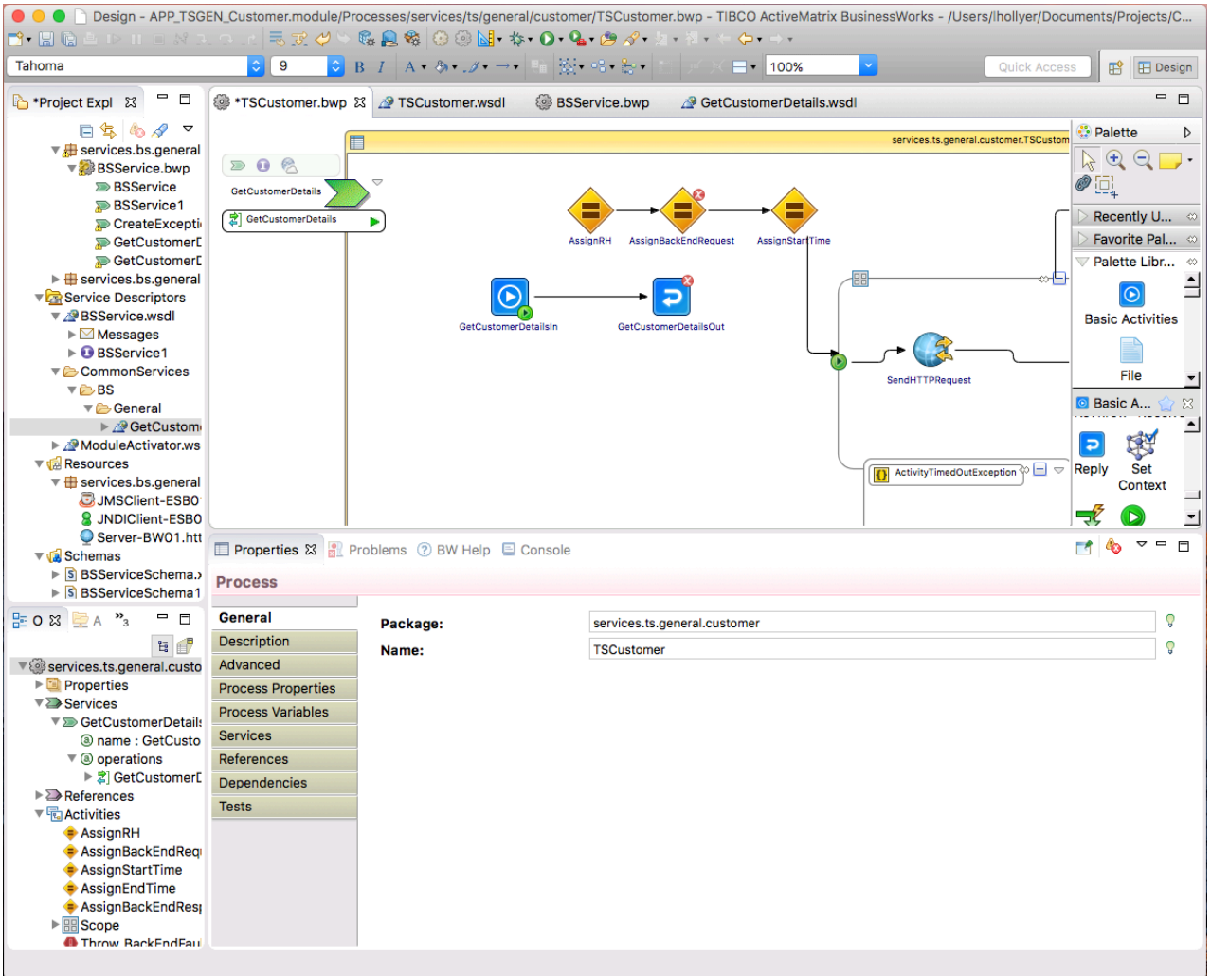


Figure 6: TS Service Provider Shared Module: Implement Service Operation

5. Create a transition between the <Operation>In task and “AssignRH” task:

N.B. if there is a conversation error on <Operation>In, click on the <Operation>In task and ensure ‘Create instance’ is checked on the general tab.

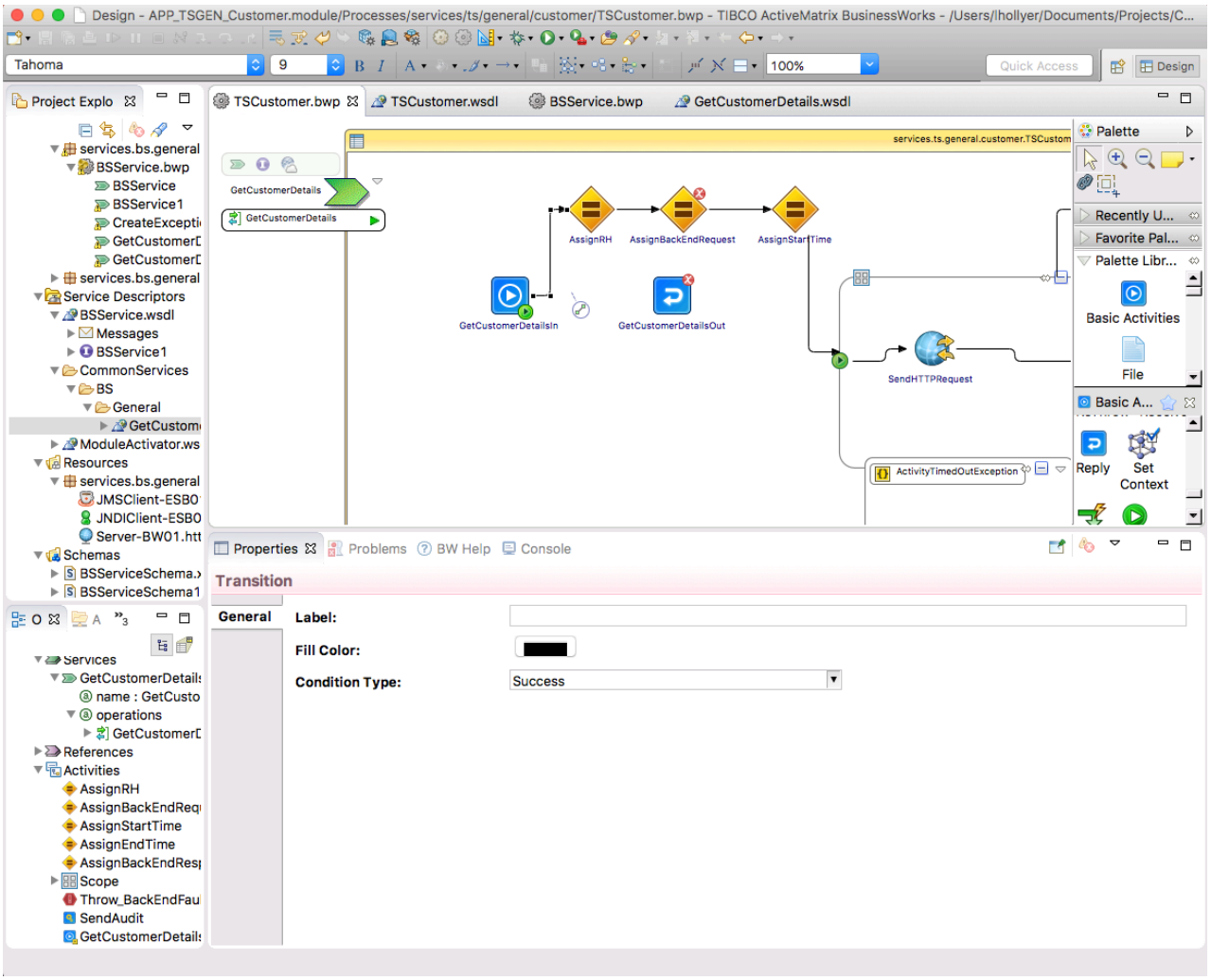


Figure 7: TS Service Provider Shared Module: Change input transition

6. Re-arrange transitions so the <Operation>Out task is located between “AssignBackEndResponse” and “SendAuditEvent” tasks:

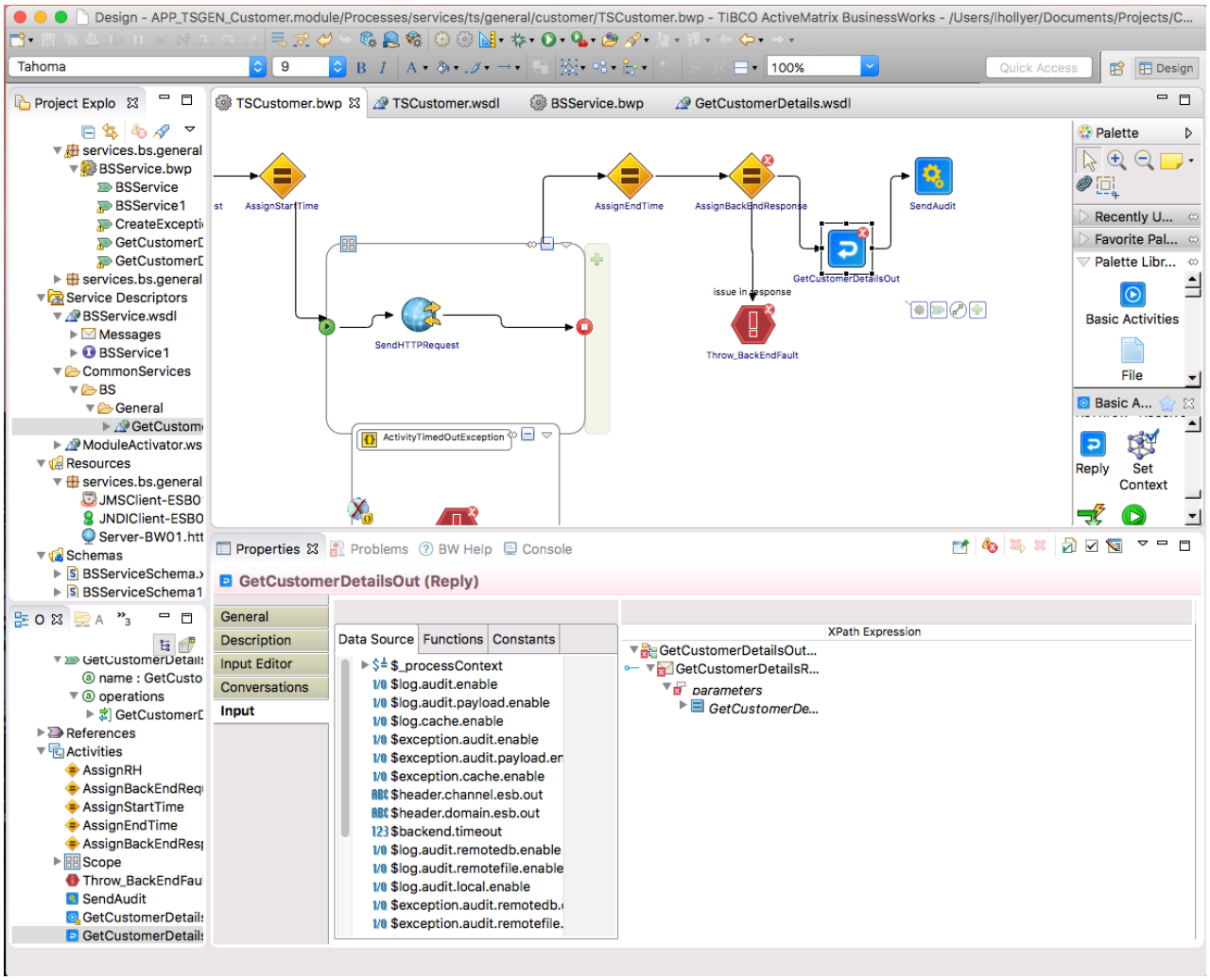


Figure 8: TS Service Provider Shared Module: Reply Transition

7. Save your changes.

6.3.5 Fix the Process Mappings

6.3.5.1 AssignRH

This task prepares the message header that can be passed into calls to other ESB services or returned in the reply or fault messages.

Fix the mapping in the task “AssignRH” as following:

1. Add a child variable called “varRequest” and change the formula for the “varRequest” variable into

```
$<Operation>In/parameters/tns9:<Operation>Request
```

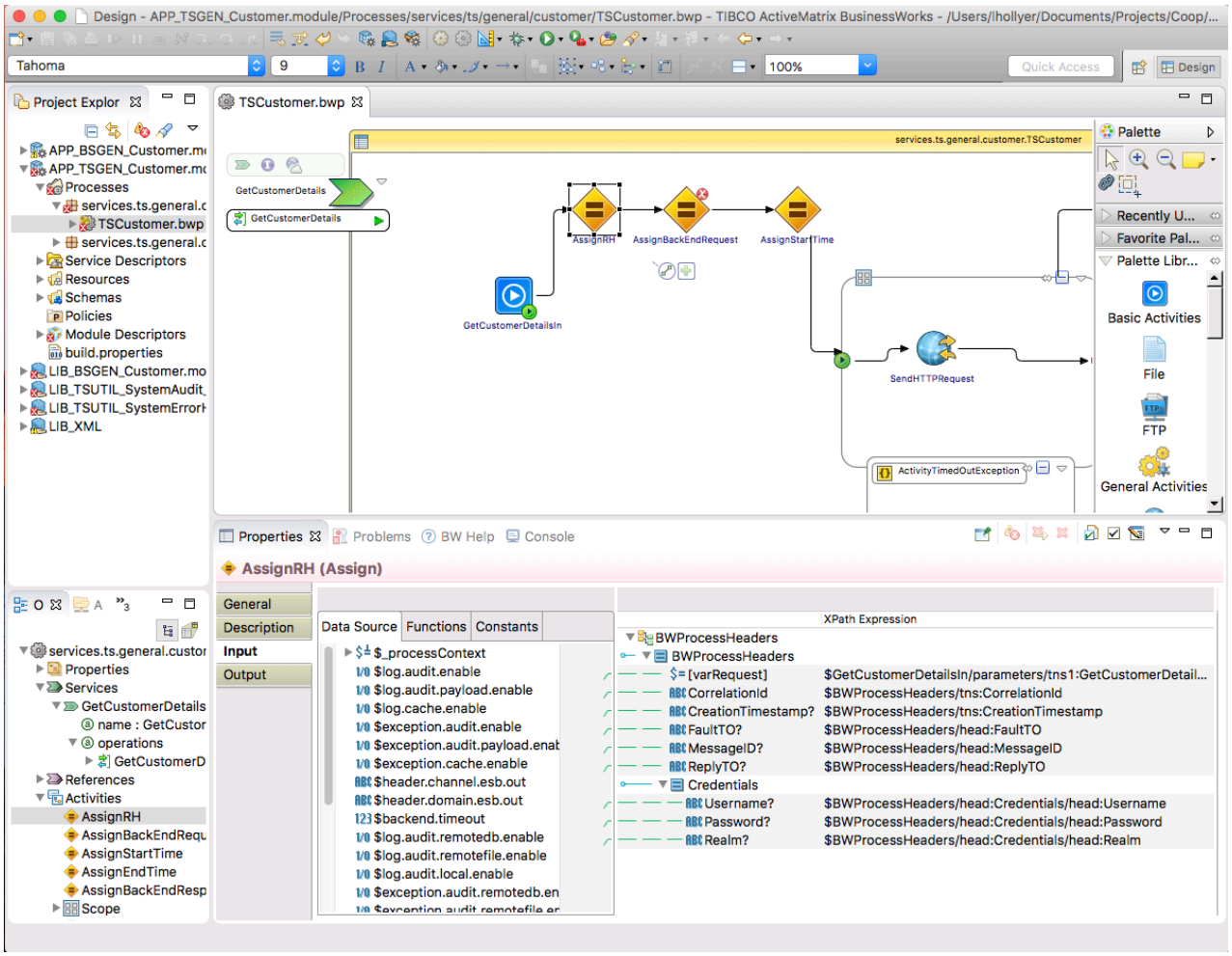


Figure 9: TS Service Provider Shared Module: AssignRH mapping

6.3.5.2 Reply_Fault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML Fault element.

Fix the configuration of the Reply_Fault task:

1. Select the service, operation and “Fault”.

Fix the mapping of the Reply_Fault task so:

1. Fault element is a copy of the CreateExceptionEvent / Response / Fault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

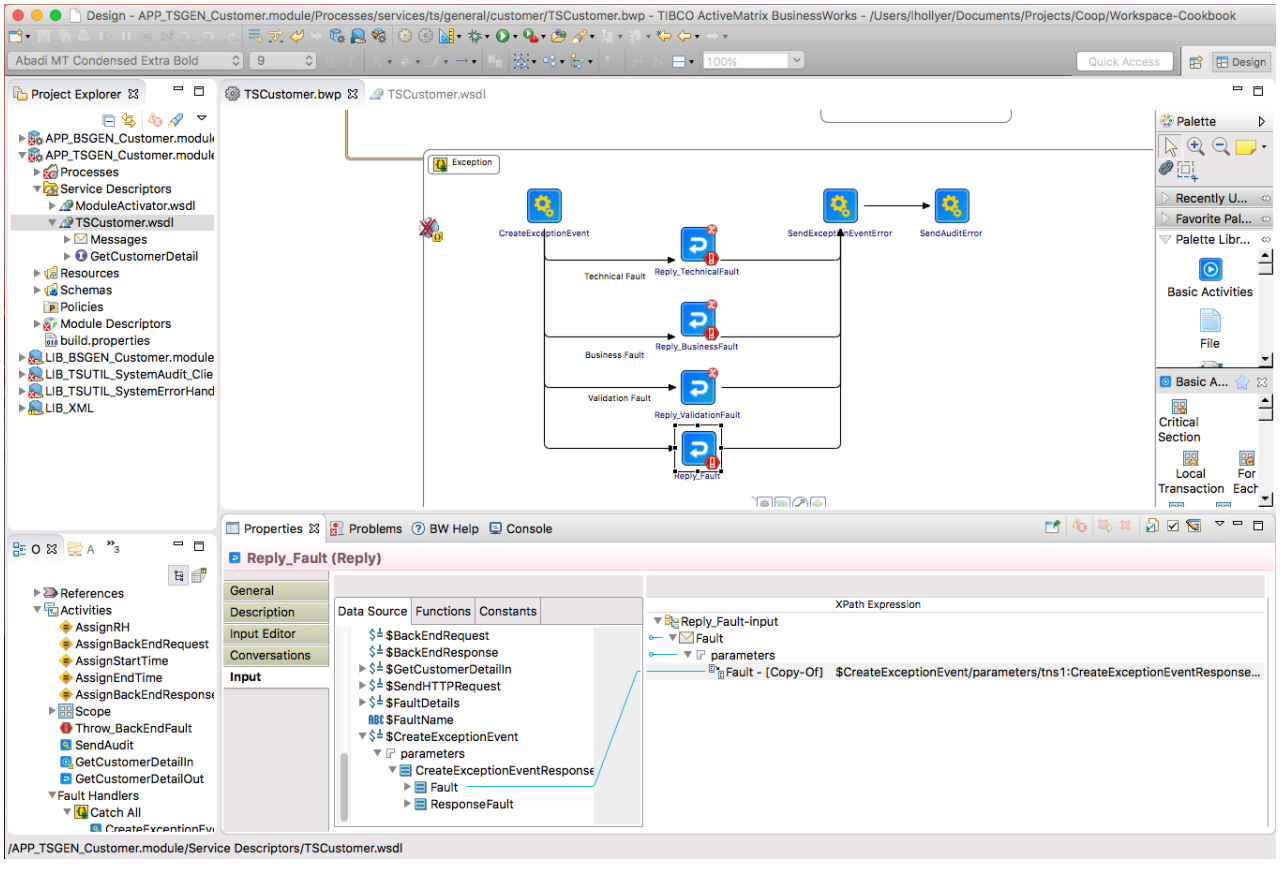


Figure 10: TS Service Provider Shared Module: Reply_Fault mapping

6.3.5.3 Reply_BusinessFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML BusinessFault element.

Fix the configuration of the BusinessFault task:

1. Select the service, operation and “BusinessFault”.

Fix the mapping of the BusinessFault task so:

1. BusinessFault element is a copy of the CreateExceptionEvent / Response / BusinessFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

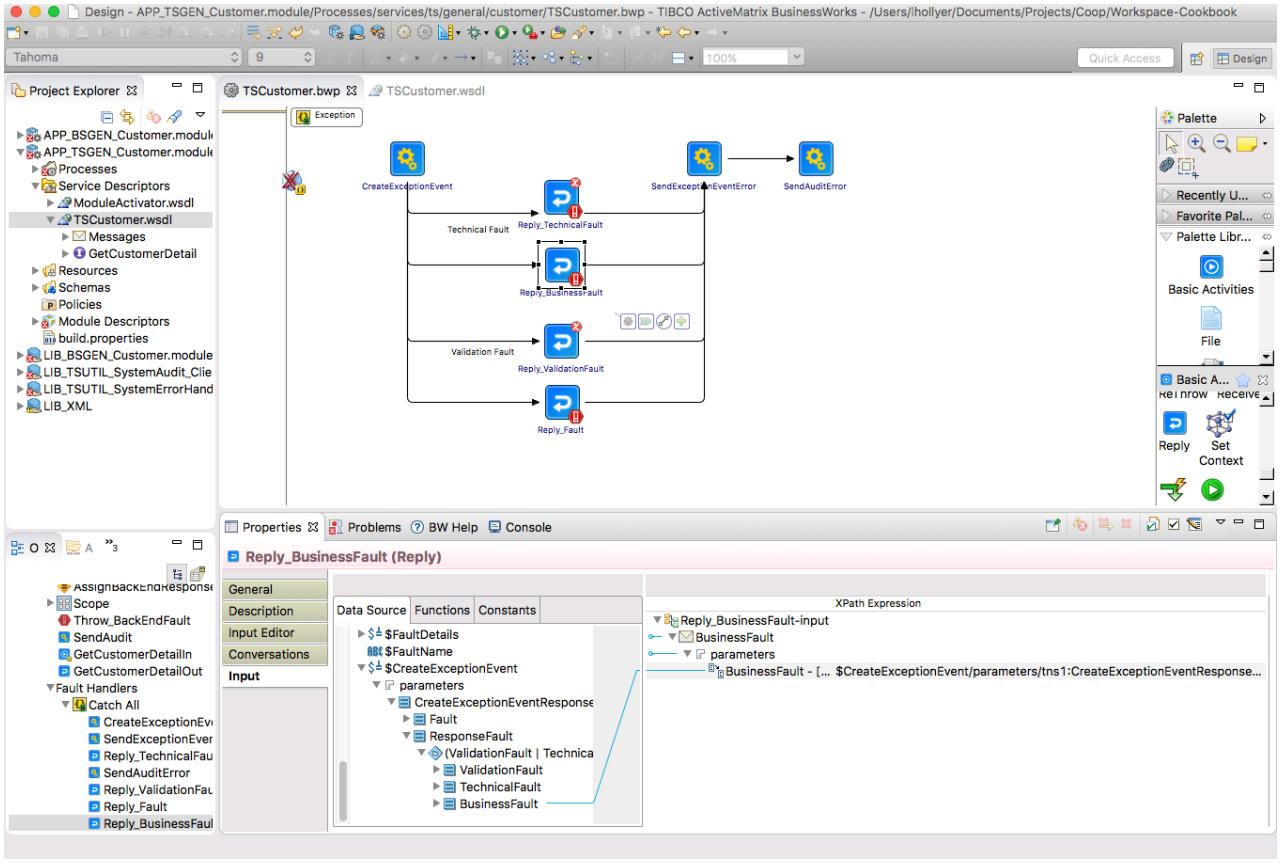


Figure 11: TS Service Provider Shared Module: Reply_BusinessFault mapping

6.3.5.4 *Reply_TechnicalFault*

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML TechnicalFault element.

Fix the configuration of the TechnicalFault task:

1. Select the service, operation and “TechnicalFault”.

Fix the mapping of the TechnicalFault task so:

1. TechnicalFault element is a copy of the CreateExceptionEvent / Response / TechnicalFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

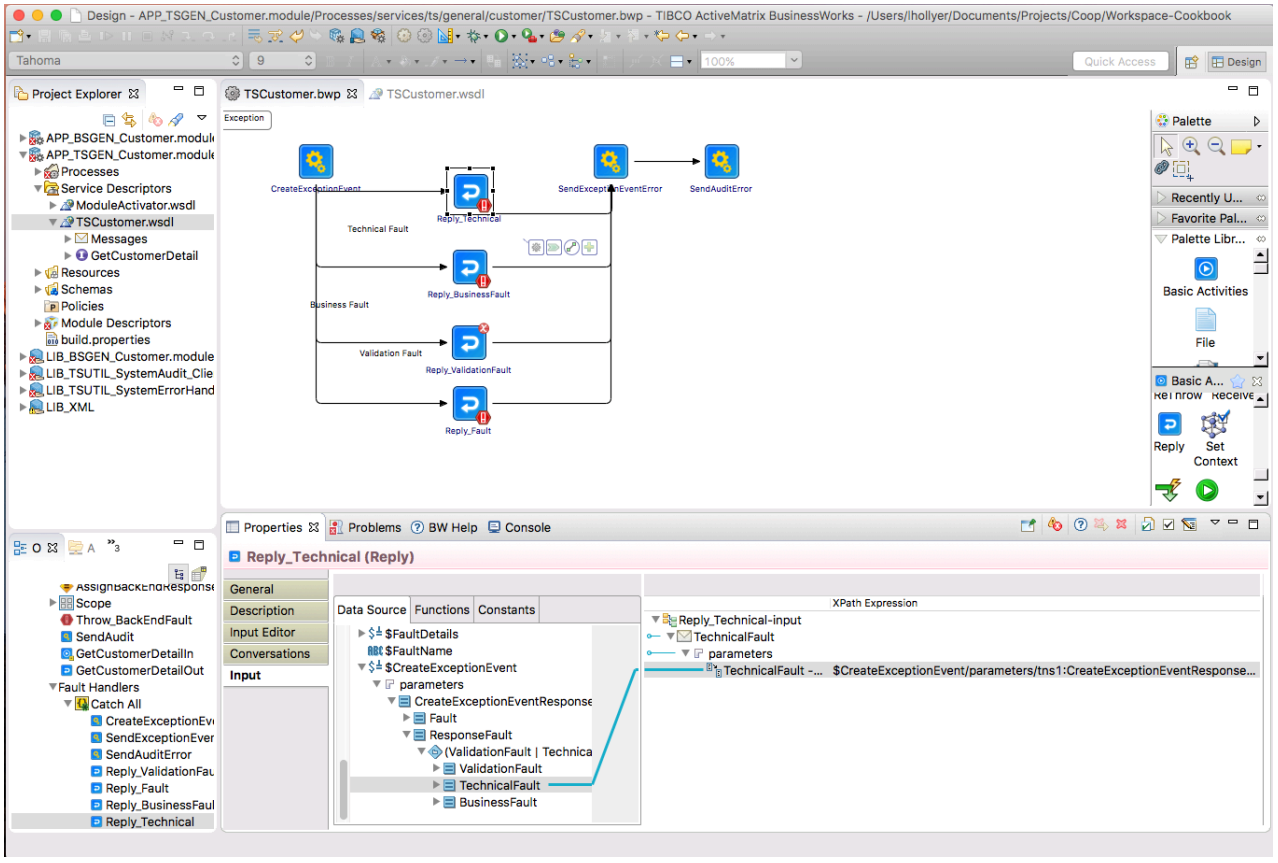


Figure 12: TS Service Provider Shared Module: Reply_TechnicalFault mapping

6.3.5.5 *Reply_ValidationFault*

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML ValidationFault element.

Fix the configuration of the ValidationFault task:

1. Select the service, operation and “ValidationFault”.

Fix the mapping of the ValidationFault task so:

1. ValidationFault element is a copy of the CreateExceptionEvent / Response / ValidationFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

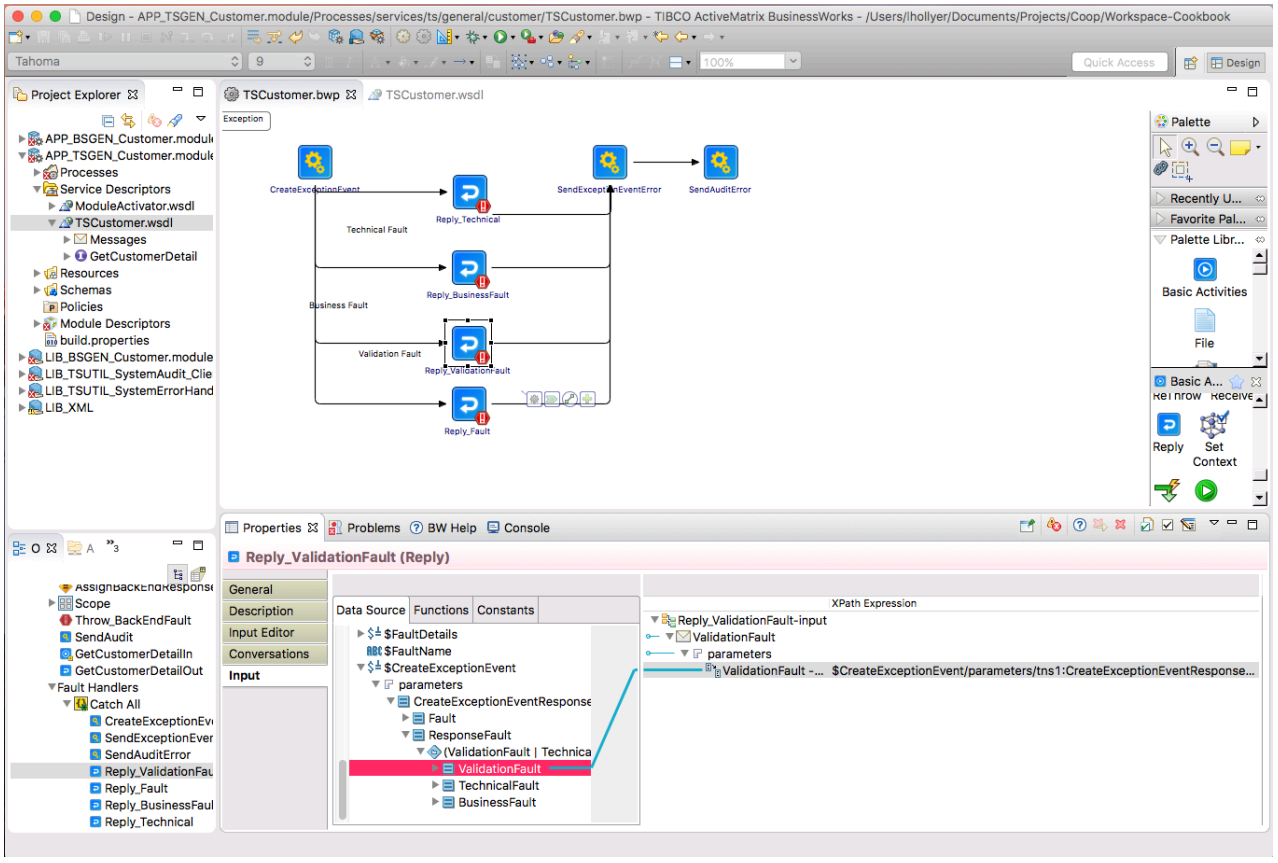


Figure 13: TS Service Provider Shared Module: Reply_ValidationFault mapping

6.3.5.6 SendExceptionEvent

This ask invokes the ESB Audit service to send one event indicating the request that was sent to the back-end and another one indicating the back-end returned a reply.

Fix the mapping of the ValidationFault task as following:

1. Add a “varBackEnd” child variable and change the value of the “varBackEnd” variable so it contains the name of the back-end system, for example “Experian”. If the back-end is .net, please use the .net service name. If the back-end is a DB you can add the name of the stored procedure as well.

6.3.5.7 SendAuditEvent_Error

This ask invokes the ESB Audit service to send one event indicating the request that was sent to the back-end, another one indicating the back-end returned a reply (if it did) and another one indicating an error happened.

Fix the mapping of the ValidationFault task as following:

1. Add a “varBackEnd” child variable and change the value of the “varBackEnd” variable so it contains the name of the back-end system, for example “Experian”. If the back-end is .net, please use the .net service name. If the back-end is a DB you can add the name of the stored procedure as well.

6.3.6 Back End on HTTP/HTTPS

If the back-end system is accessed on HTTP/HTTPS transport, you can modify the service as following.

6.3.6.1 Change the resources

1. Rename the HTTP Client resource from “services.ts.backend.servicename.Client-BACKEND-http” into “services.ts.<area>.<sevicename>.Client-<BackEndName>”.
2. Delete the JDBC resource “Client-BACKEND-jdbc”.
3. Delete the old resource package services.ts.backend.servicename.

6.3.6.2 Change the Module Properties

1. Go to the module properties.
2. Select the group “services.ts.<area>.<servicename> / <ServiceName> / backend”. In the properties pane, use the light-bulb icon to rename the group into “<BackEndName>”.

   Example:

   services.ts.<area>.<servicename> / <ServiceName> / backend / becomes
   services.ts.bline.getcardscheme / GetCardScheme / BLINE
3. In the group you renamed, change the “endpoint.uri” value to correspond to the back-end URI.
4. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”.
5. Select the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”. In the properties pane, use the light-bulb icon to rename the group into “Client-<BackEndName>”.
6. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-jdbc”.

6.3.6.3 Change Service Process (if back-end is accessed via SOAP)

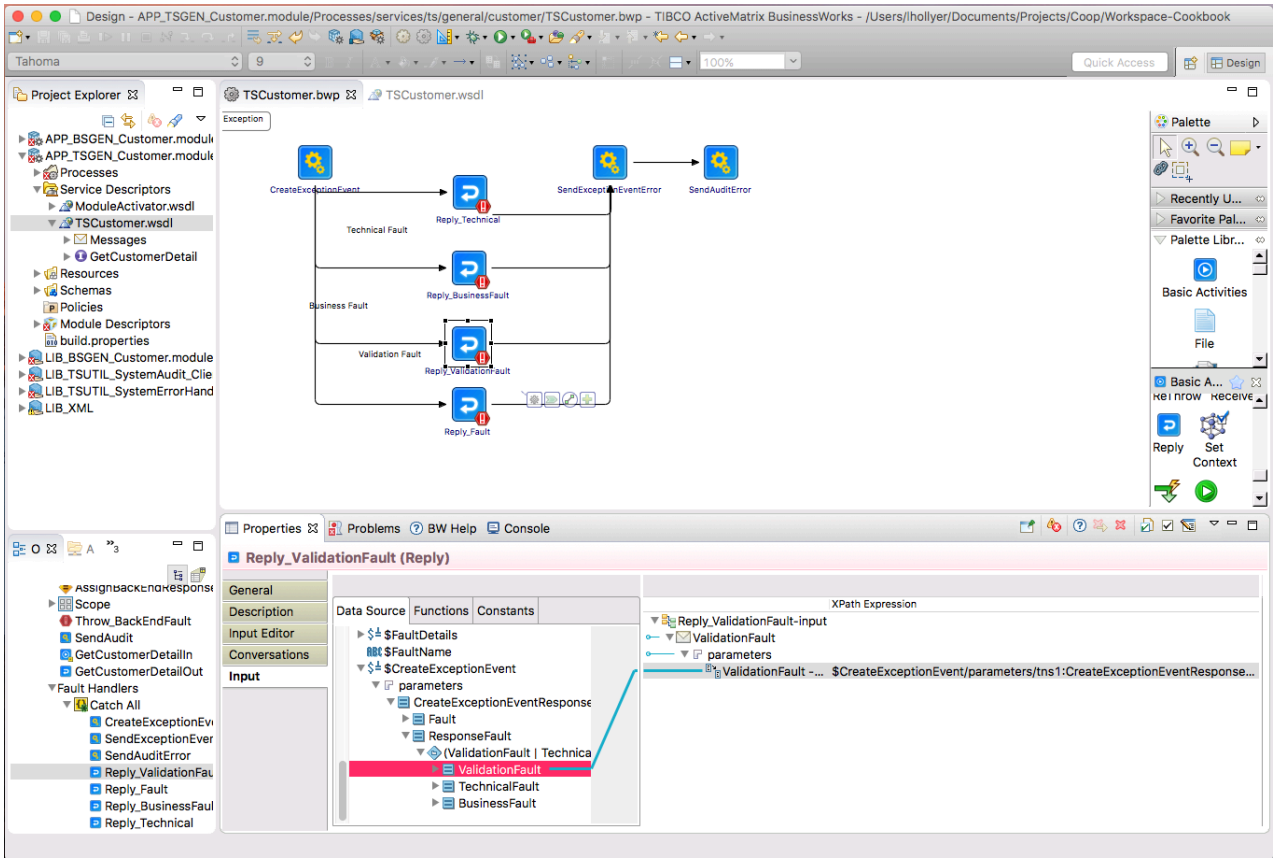
1. Import the concrete WSDL into the “Service Descriptors” folder.

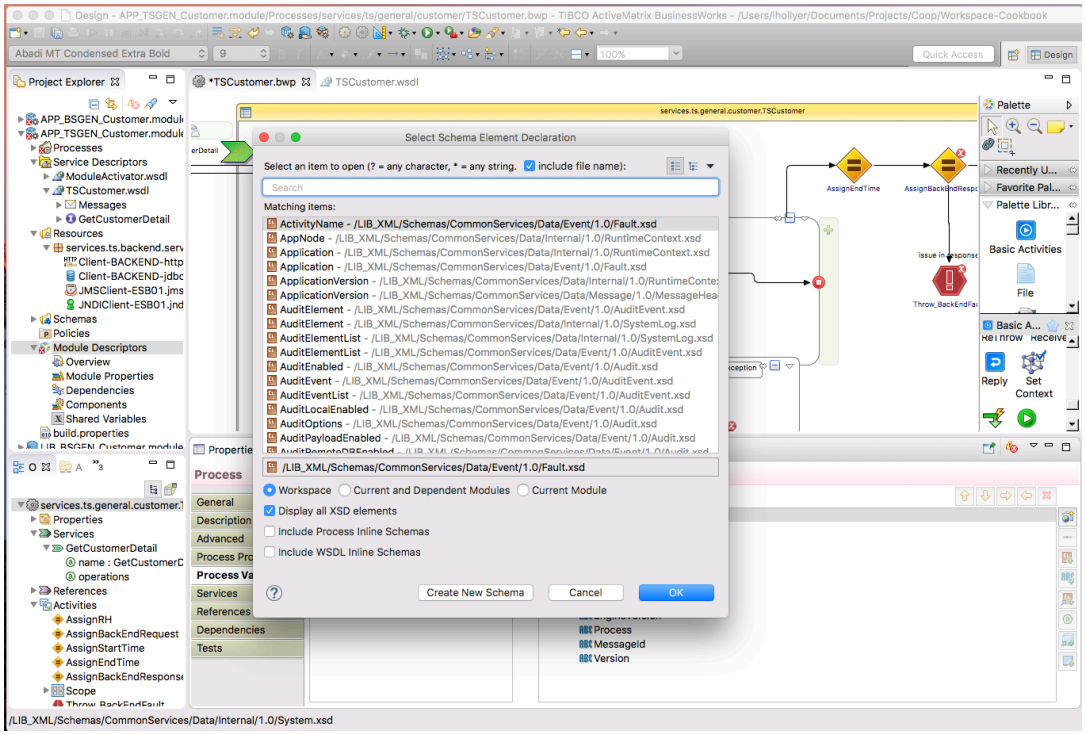
Change the service process as following:

Service Invocation

1. Remove the “SendHTTPRequest” task and replace it with an Invoke task.
2. Configure the Service Reference and add a SOAP binding which uses:
   a. The HTTP client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.
   b. The property “services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / endpoint.uri.” for the endpoint URI.

3. Edit the process variable “$BackEndRequest” so is uses the XML schema of the back-end operation request (from its concrete WSDL).





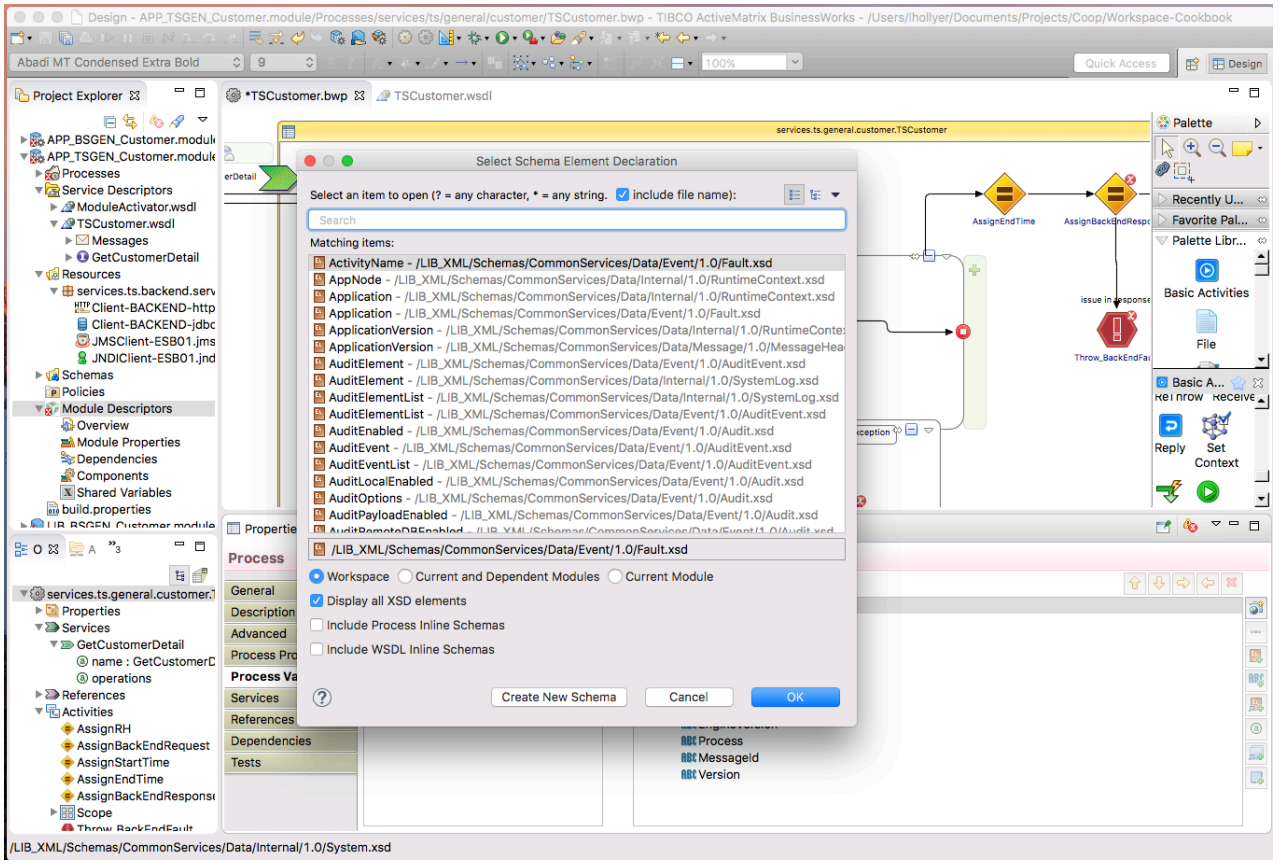
4. Edit the process variable “$BackEndResponse” so is uses the XML schema of the back-end operation response (from its concrete WSDL).
5. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).
6. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.
7. Map the response from the back-end in the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).

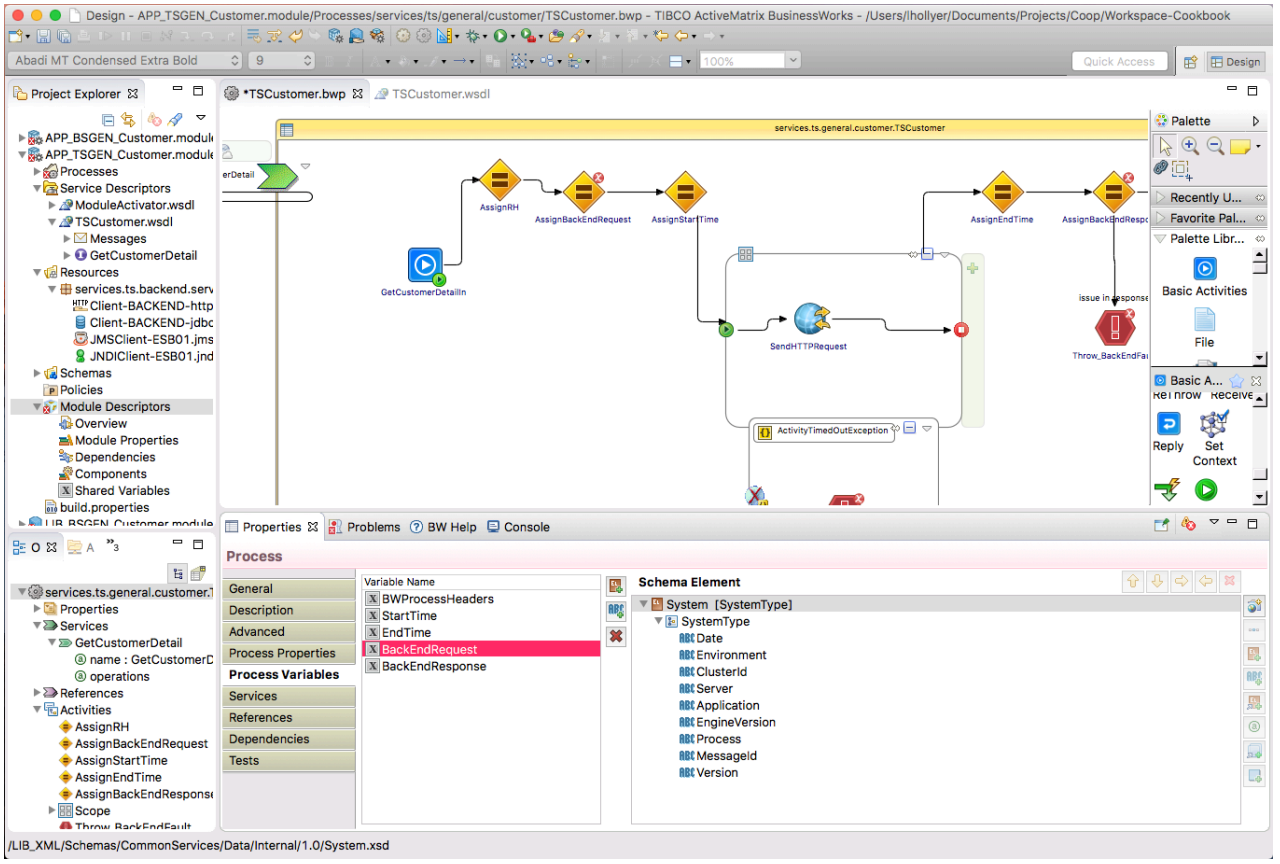
6.3.6.4 Change Service Process (if back-end is accessed via raw HTTP)

Change the service process as following:

Service Invocation

1. Edit the process variable “$BackEndRequest” so is uses the XML schema of the back-end operation request.


Select Schema Element Declaration
Select an item to open (? = any character, * = any string. include file name):
Search
Matching items:
ActivityName - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
AppNode - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeContext.xsd
Application - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeContext.xsd
Application - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
ApplicationVersion - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeConte
ApplicationVersion - /LIB_XML/Schemas/CommonServices/Data/Message/1.0/MessageHea
AuditElement - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditElement - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/SystemLog.xsd
AuditElementList - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/SystemLog.xsd
AuditElementList - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditEvent - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditEventList - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditLocalEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditOptions - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditPayloadEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
/LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
Workspace
Current and Dependent Modules
Current Module
Display all XSD elements
Include Process Inline Schemas
Include WSDL Inline Schemas
Create New Schema
Cancel
OK
/LIB_XML/Schemas/CommonServices/Data/Internal/1.0/System.xsd



2. Edit the process variable “$BackEndResponse” so is uses the element “Payload” from the **LIB_XML / Schemas / CommonServices / Data / Event / 1.0 / AuditEvent.xsd** XSD.

3. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).

4. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.

5. Map the $SendHTTPRequest variable into the input the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).

6.3.6.5 Change Service Process (all cases)

Detect Back End Errors

If the service must analyze the back-end reply to detect back-end errors, this can be done in the “Throw_BusinessFault” task and the transition from “AssignBackEndResponse” task to “Throw_BusinessFault” task. In the “Throw_BusinessFault” task you can change the text and description of the issue.

If not, you can delete the “Throw_BusinessFault” task.

If the service operation can throw various SOAP faults that can be mapped to an ESB BusinessFault, you can handle them as following:

5. Right-click on the Invoke task and in the contextual menu, select the option “Catch / <fault>” where <fault> if the SOAP fault to handle.

6. In the new Catch block, add a ”Throw” task.

7. In “Input Editor” of the “Throw” task, add the XML element “ThrowableBusinessFault” from the “Schemas / CommonServices / Data / Message / Internal / Throwable” XSD.

8. Change the mapping of the task to populate the text and description of the fault.

Timeout Faults

1. In the “Throw_TimeoutFault” task, you can change the text and description of the timeout error.

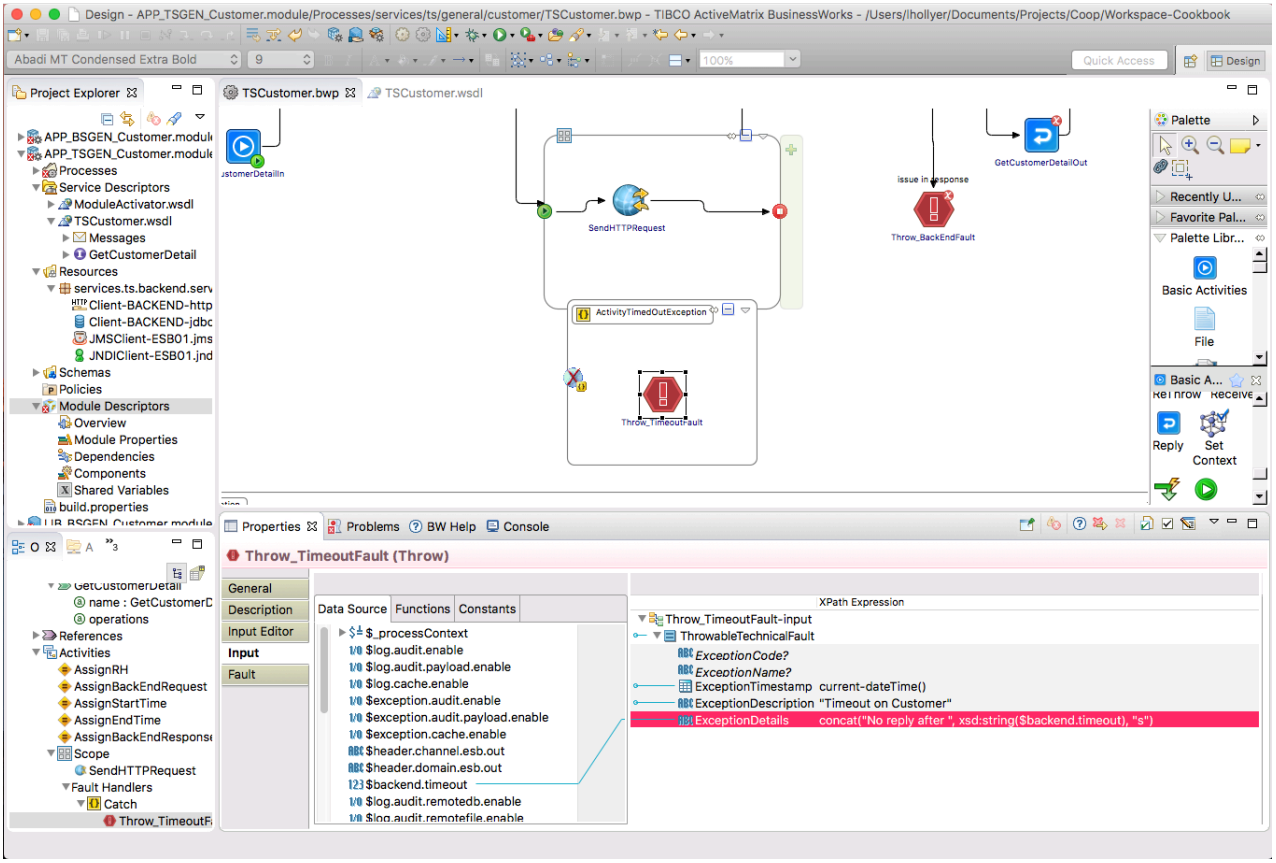


Figure 14: TS Service Provider Shared Module: Throw_TimeoutFault mapping

2. If the task has no input, fix it as following: in “Input Editor” add the XML element “ThrowableBusinessFault” from the “Schemas / CommonServices / Data / Internal / 1.0 / Internal / Throwable” XSD:

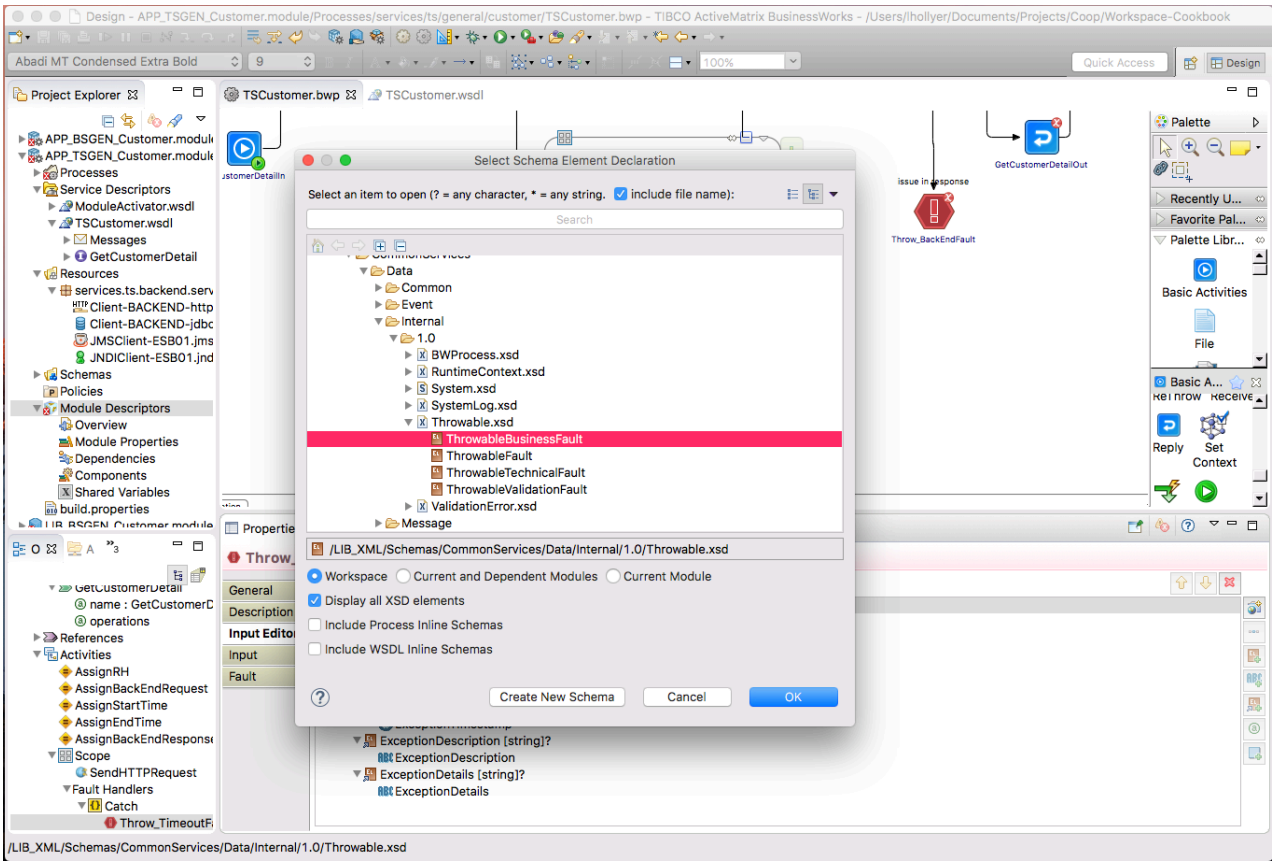


Figure 15: TS Service Provider Shared Module: Throw_TimeoutFault configuration

3. Delete and recreate the task if necessary.

6.3.6.6 Add HTTPS Resources

If the back-end is accessed via HTTPS, the following resources must be added manually and named as such:

- Resources:
    - SSL Client Configuration: services.ts.<area>.<sevicename>.Client-<BackEndName>-ssl
    - Keystore Provider Resource: services.ts.<area>.<sevicename>.Client-<BackEndName>-jks
- Module Properties
    - Keystore URL: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.url**.
    - Keystore Password: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.password**.
    - Refresh Interval: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.refresh.interval** (3600000 ms as default value)

6.3.7 Back End on JDBC

If the back-end system is a database, you can modify the service as following.

6.3.7.1 Change the resources

1. Rename the JDBC Client resource from “services.ts.backend.servicename.Client-BACKEND-jdbc” into “services.ts.<area>.<servicename>.Client-<BackEndName>”.
2. Delete the HTTP resource “Client-BACKEND-http”.
3. Delete the old resource package services.ts.backend.servicename.

6.3.7.2 Change the Module Properties

1. Go to the module properties.
2. Select the group “services.ts.<area>.<servicename> / <ServiceName> / backend / BACKEND”. In the properties pane, use the light-bulb icon to rename the group into “<BackEndName>”.
3. In the group you renamed, delete the “endpoint.uri” property.
4. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”.
5. Select the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-jdbc”. In the properties pane, use the light-bulb icon to rename the group into “Client-<BackEndName>”.
6. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”.

6.3.7.3 Change Service Process

Change the service process as following:

1. Remove the “SendHTTPRequest” task and replace it with a JDBC task.
2. Configure the JDBC tasks so it uses:
    a. The JDBC client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.
    b. The property “services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / timeout” for the timeout.
3. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).
4. Map the response from the back-end in the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).
5. If the service must analyze the back-end reply to detect back-end errors, this can be done in the “Throw_BusinessFault” task and the transition from “AssignBackEndResponse” task to “Throw_BusinessFault” task. In the “Throw_BusinessFault” task you can change the text and description of the issue.

    If not, you can delete the “Throw_BusinessFault” task.

6. In the “Throw_TimeoutFault” task, you can change the text and description of the timeout error.

6.4 Implement the Service

6.4.1 Service Response

In the <Operation>Out task, you MUST map the ResponseHeader element as following:

- **CorrelationId**: $BWProcessHeaders / CorrelationId
- **MessageID**: $BWProcessHeaders / MessageID
- **Version**: $BWProcessHeaders / Response / Version
- **ResponseDateTime**: current-dateTime()

N.B. if <Operation>Out does not have the ResponseHeader element, check the WSDL location imports the Schema in LIB_TS<Service>.module's XSD, and check the XSD has the ResponseHeader set to ResponseHeaderType from LIB_XML.

6.4.2 Invoking another ESB service

If you invoke another ESB service (BS or TS), you MUST map the RequestHeader element as following:

- **CorrelationId**: $BWProcessHeaders / CorrelationId
- **MessageID**: $BWProcessHeaders / MessageID
- **Version**: The version of the invoked service, for example "1.0".
- **RequestDateTime**: current-dateTime()

N.B. if <Operation>Out does not have the ResponseHeader element, check the WSDL location imports the Schema in LIB_TS<Service>.module's XSD, and check the XSD has the RequestHeader set to RequestHeaderType from LIB_XML.

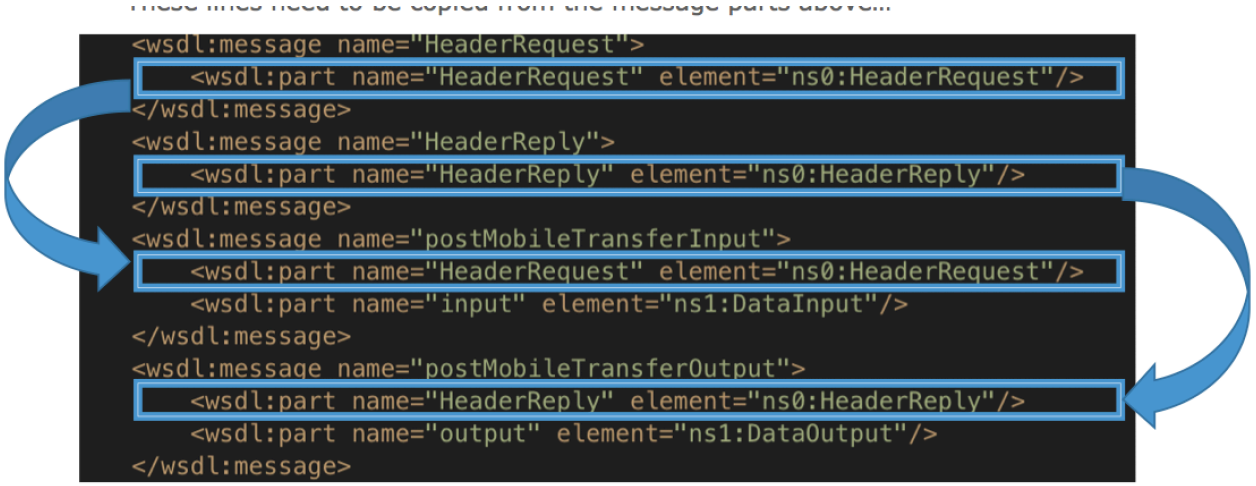
6.4.3 Validate the Request

Some service may require additional input data validation (besides what is defined inside the service schema, which BW validates by default). In such case, you must implement the validation as following:

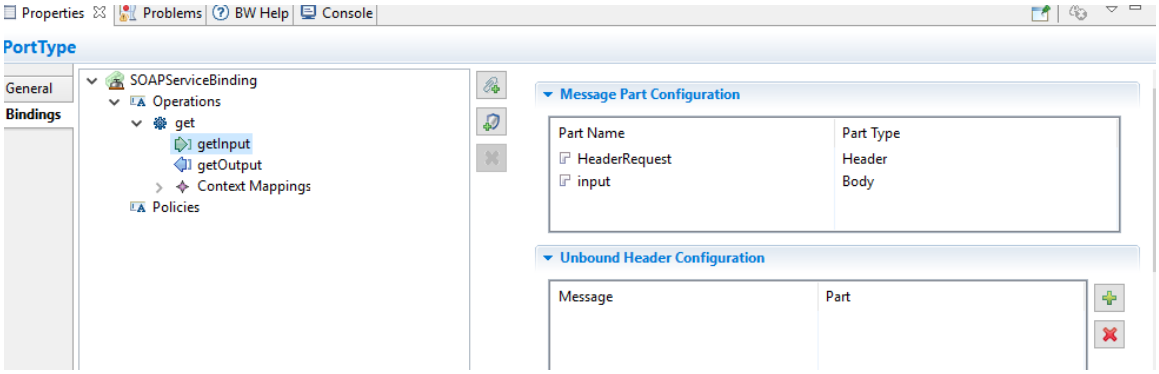
- The steps are the same as for a business service (see section 4.3.5), but the "Throw_ValidationFault" happens between the "AssignRH" task and "AssignBackEndRequest" task.

6.5 Using an Existing Concrete WSDL

1. Duplicate the lines highlighted in the concrete WSDL file:



2. Follow the previous steps as normal to import the WSDL into the process.
3. In Module Descriptors / Components, select the binding and click on “Bindings” in the properties pane.
4. Change the input operation HeaderRequest to be of type “Header” instead of “Body”:



5. Change the output operation HeaderResponse to be of type “Header” too.
6. Continue through the steps as normal.

7 Create a Technical Service Provider as an Application

This chapter explains how to create a new Technical Service exposed on a SOAP over JMS binding, on a JMS queue.

7.1 Template Overview

The template module “APP_ServiceTemplate.module” (renamed as APP_TSServiceTemplate.module) allows you to create a new application for a Business service. It contains the following objects:

- Processes:
  - **TSService**: the service implementation.
  - **Activator**: the process executed when the application starts up inside an appnode. This can be used to trace configuration settings (such as back-end URLs) or call other Activator required in dependent modules.
- WSDLs:
  - **ModuleActivator.wsdl**: the abstract WSDL of the **Activator** process (this one should never be changed at all: it is created by BW when the Activator process is created).
- Resources:
  - **Client-BACKEND-http:** the HTTP client the Technical service MUST use if the back-end system is exposed on HTTP/HTTPS transport.
  - **Client-BACKEND-jdbc:** the JDBC client the Technical service MUST use if the back-end system is exposed as a database.
  - **JNDIClient-ESB01**: the JNDI connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.
  - **JMSClient-ESB01**: the JMS connection to the ESB Server that the Business service MUST use when invoking other ESB services exposed on SOAP over JMS.

Notes:

4 The JNDI and JMS connections used by the System Error Handler and System Audit are separated from JNDIClient-ESB01 and JMSClient-ESB01. They are contained in the LIB_TSUTIL_SystemErrorHandler_Client and LIB_TSUTIL_SystemAudit_Client modules.

7.2 Pre-Requisites

The XSD and WSDL have been created.

7.2.1 Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “APP_TSServiceTemplate.module” into the folder **<GIT> / trunk / BW / TechnicalServices**.

2. Rename the copied folder into “APP_TS<Area>_<ServiceName>”.
3. With a text editor, open the .project file into the copied folder and change the project name at the top from APP_TSServiceTemplate.module into APP_TS<Area>_<ServiceName>.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>APP_TSCMD_GetVehicleDetails</name>
	<comment></comment>
…
```

With TIBCO BusinesStudio

4. Start TIBCO BusinessStudio
5. In your workspace, import the following modules:
    a. LIB_XML (from <GIT> / trunk / XML
    b. LIB_TSUTIL_SystemAudit_Client (from <GIT> / trunk / BW / TechnicalServices)
    c. LIB_TSUTIL_SystemErrorHandler_Client (from <GIT> / trunk / BW / TechnicalServices)
    d. LIB_TS<Area>_<ServiceName> (from <GIT> / trunk / BW / TechnicalServices)

All subsequent changes are done on the APP_<TYPE><Area>_<ServiceName> module:

5. Go to the module overview.
6. Change the name into “APP_<TYPE><Area>_<ServiceName> Module”:

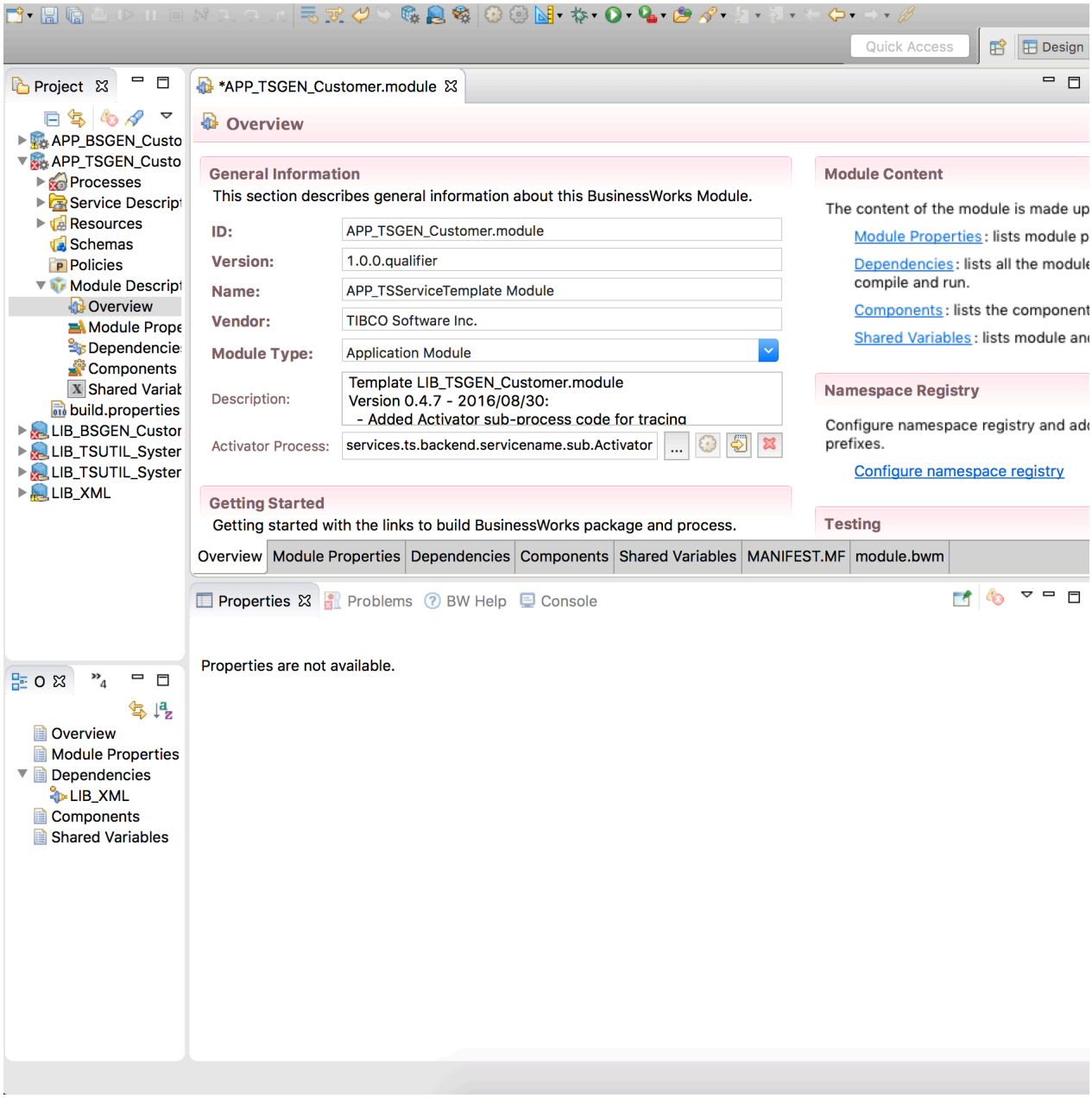


Figure 16: TS Service Provider Shared Module: Module Name

7.2.2 Change the Module Properties

5. Go to the module properties.

6. Select the group “services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”:

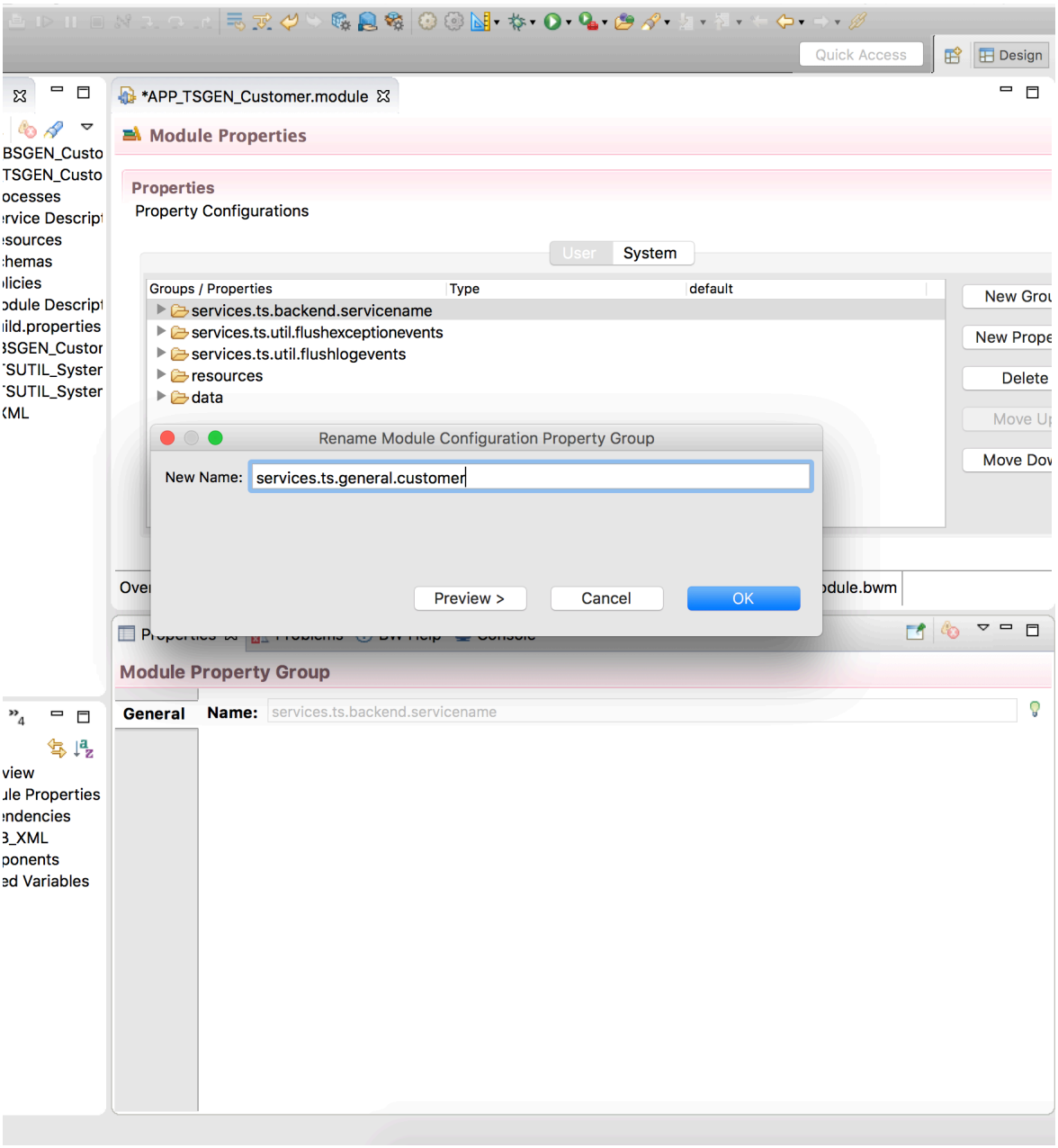
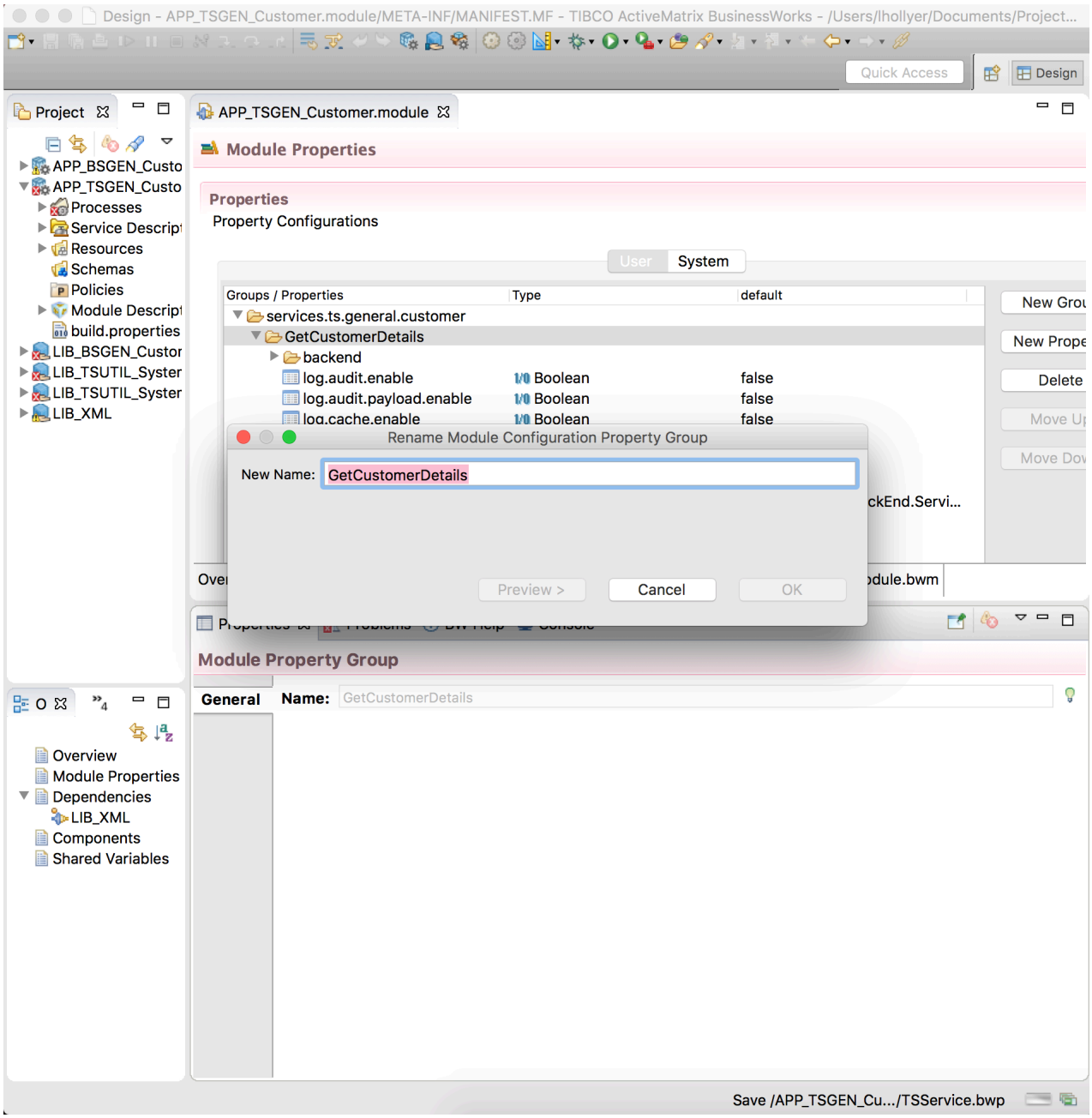


Figure 17: TS Service Provider Shared Module: Module Properties

7. In this group, use the same technique to rename the group “TSService” into “<ServiceName>” (for example “GetCustomerDetails”).



8. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”:

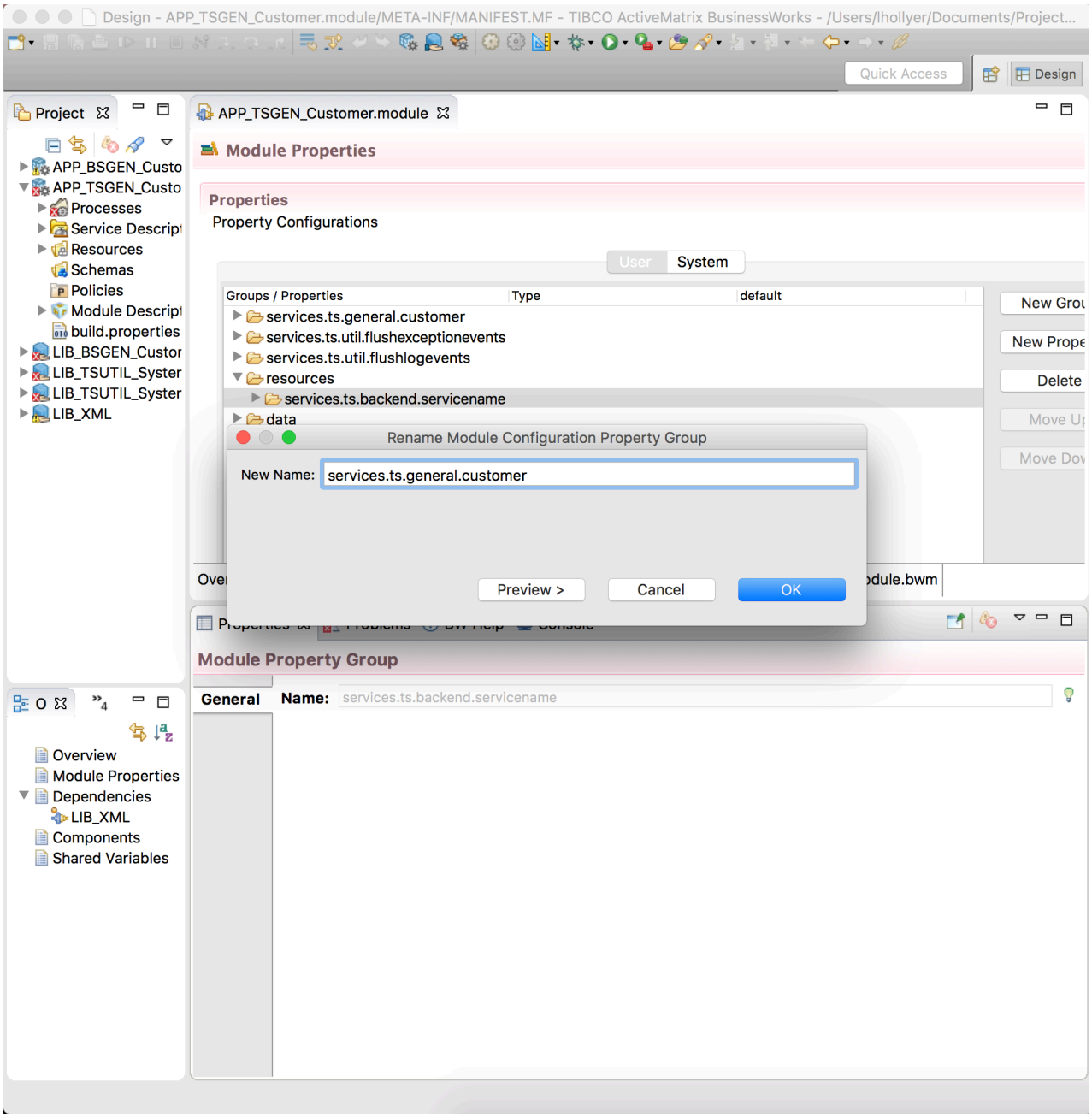


Figure 18: TS Service Provider Shared Module: Resource Properties

7.2.3 Create the Process Packages

6. Create the process package “services.ts.<area>.<servicename>”.
7. Move the TSService process into it.
8. Create the process package “services.<type>.<area>.<servicename>.sub”.
9. Move the Activator process into it.

10. Delete the remaining …servicename… process packages.

7.2.4 Change the Service Process

8. Open the process “services.ts.<area>.<servicename> / TSService”.

9. In the properties, use the light-bulb icon to rename it into “<ServiceName>”:

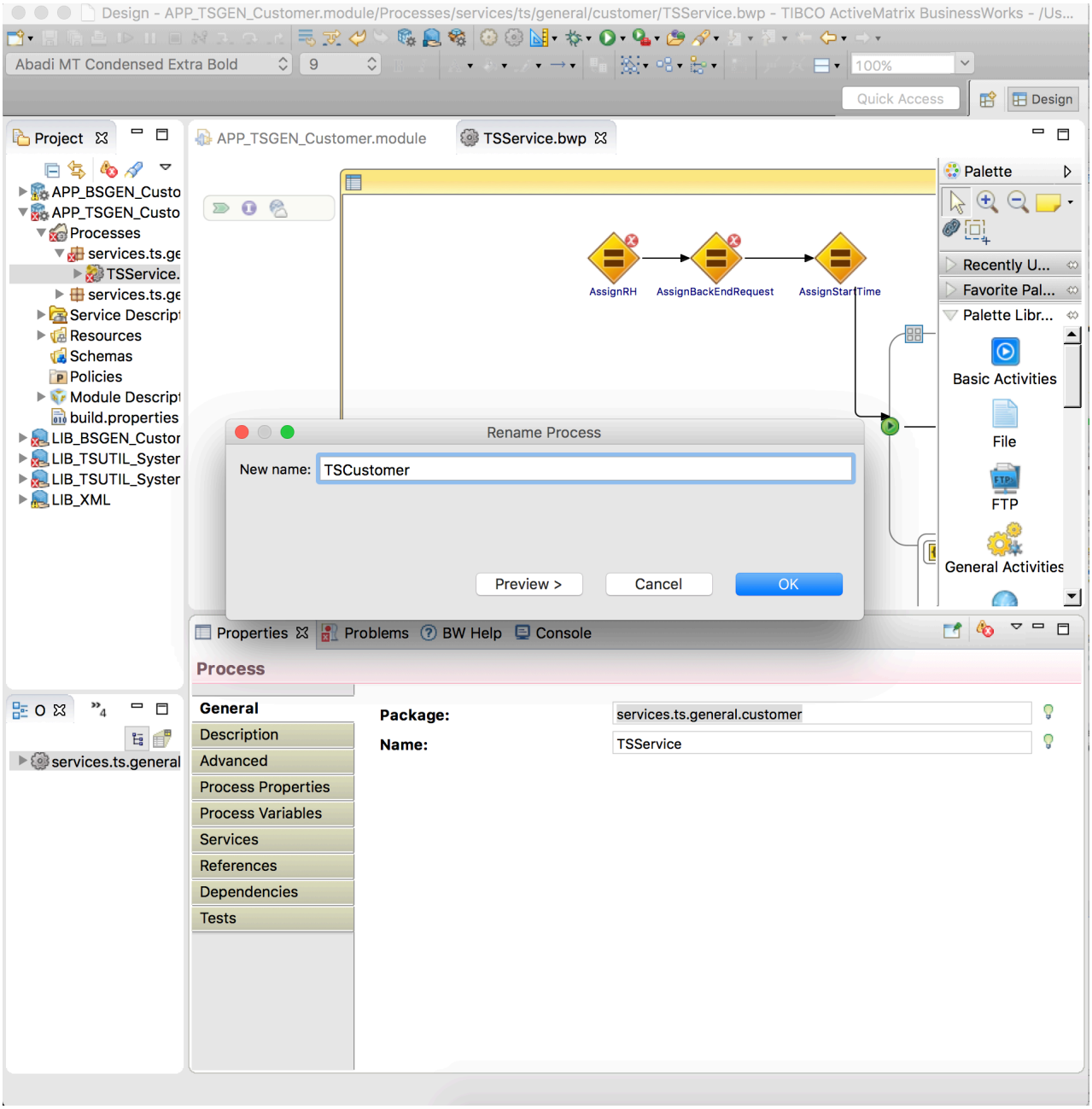


Figure 19: TS Service Provider Shared Module: Process Name

10. Import the XSD and service WSDL of the TS Service into the module.

11. Drag and drop the abstract WSDL into the process:

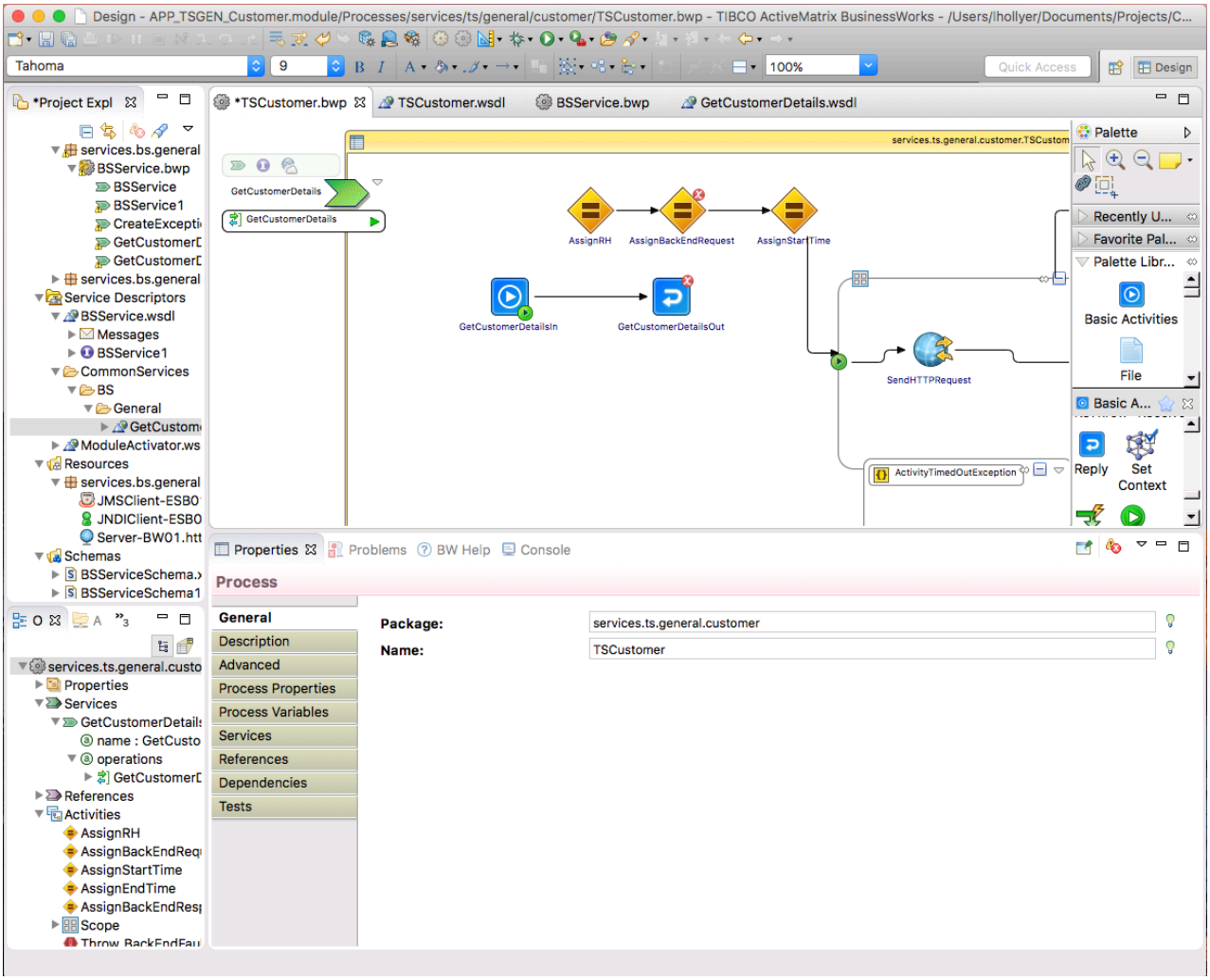


Figure 20: TS Service Provider Shared Module: Implement Service Operation

12. Create a transition between the <Operation>In task and “AssignRH” task:

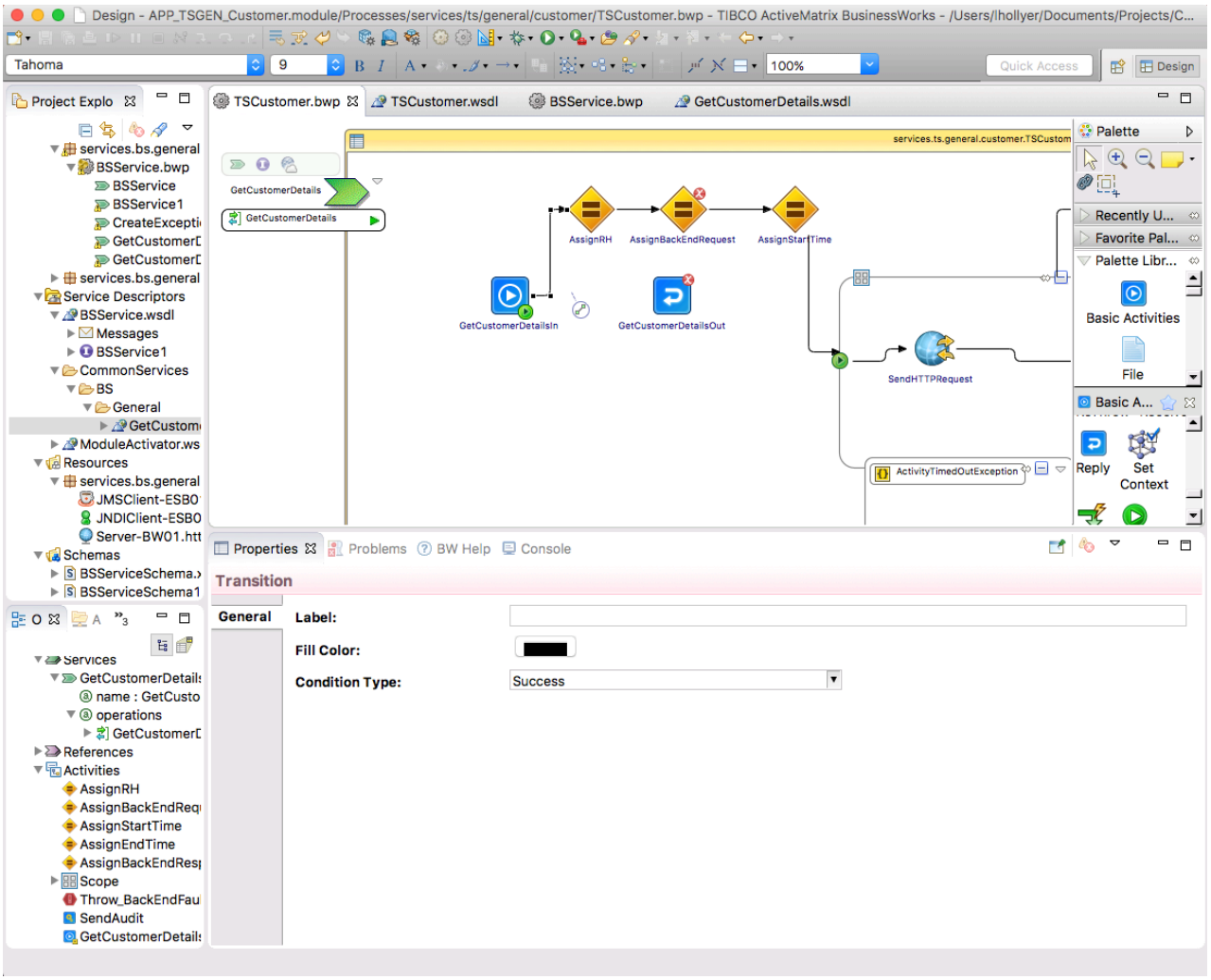


Figure 21: TS Service Provider Shared Module: Change input transition

13. Re-arrange transitions so the <Operation>Out task is located between “AssignBackEndResponse” and “SendAuditEvent” tasks:

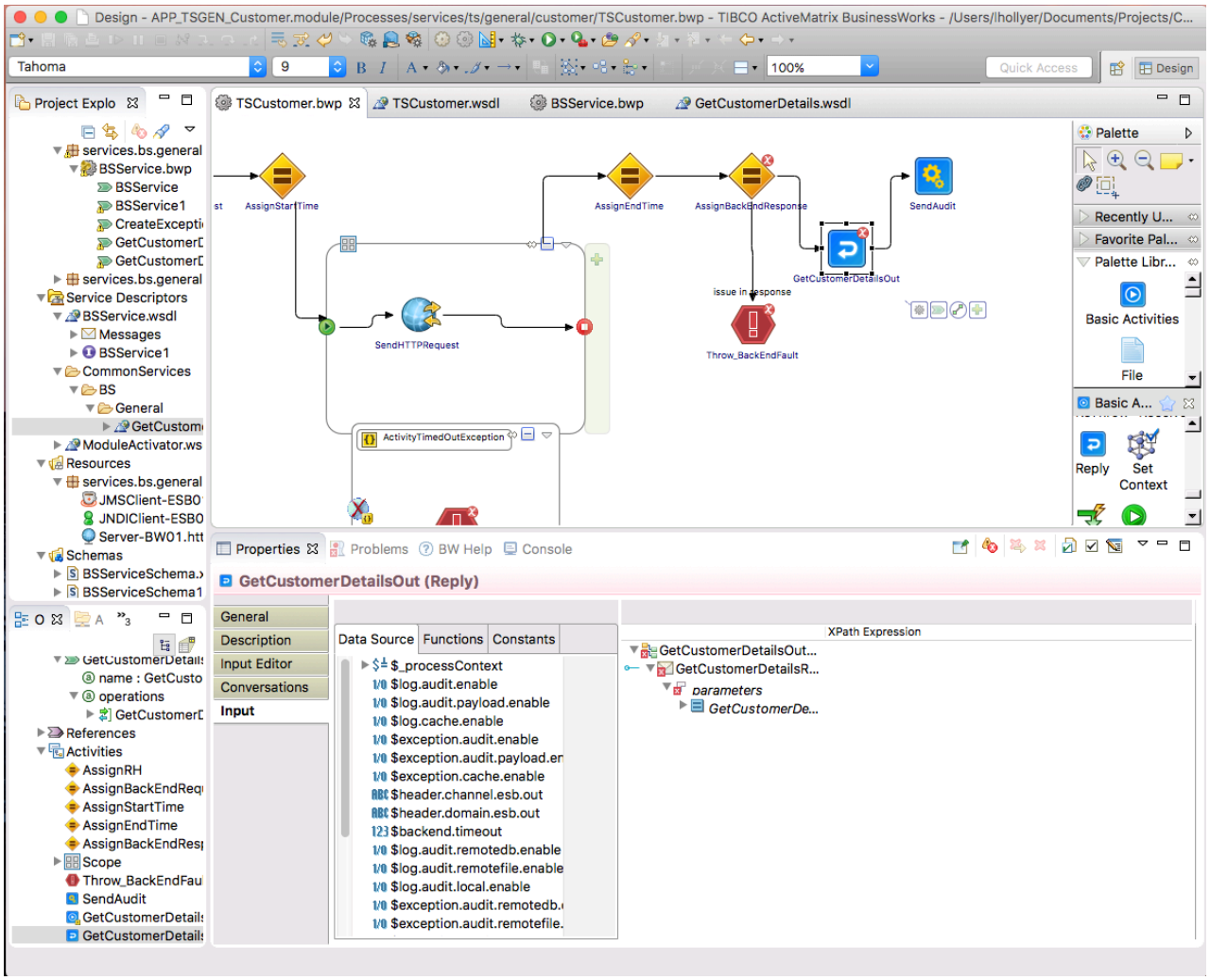


Figure 22: TS Service Provider Shared Module: Reply Transition

14. Save your changes.

7.2.5 Fix the Process Mappings

7.2.5.1 AssignRH

This task prepares the message header that can be passed into calls to other ESB services or returned in the reply or fault messages.

Fix the mapping in the task “AssignRH” as following:

2. Add a child variable called “varRequest” and change the formula for the “varRequest” variable into

```
$<Operation>In/parameters/tns9:<Operation>Request
```

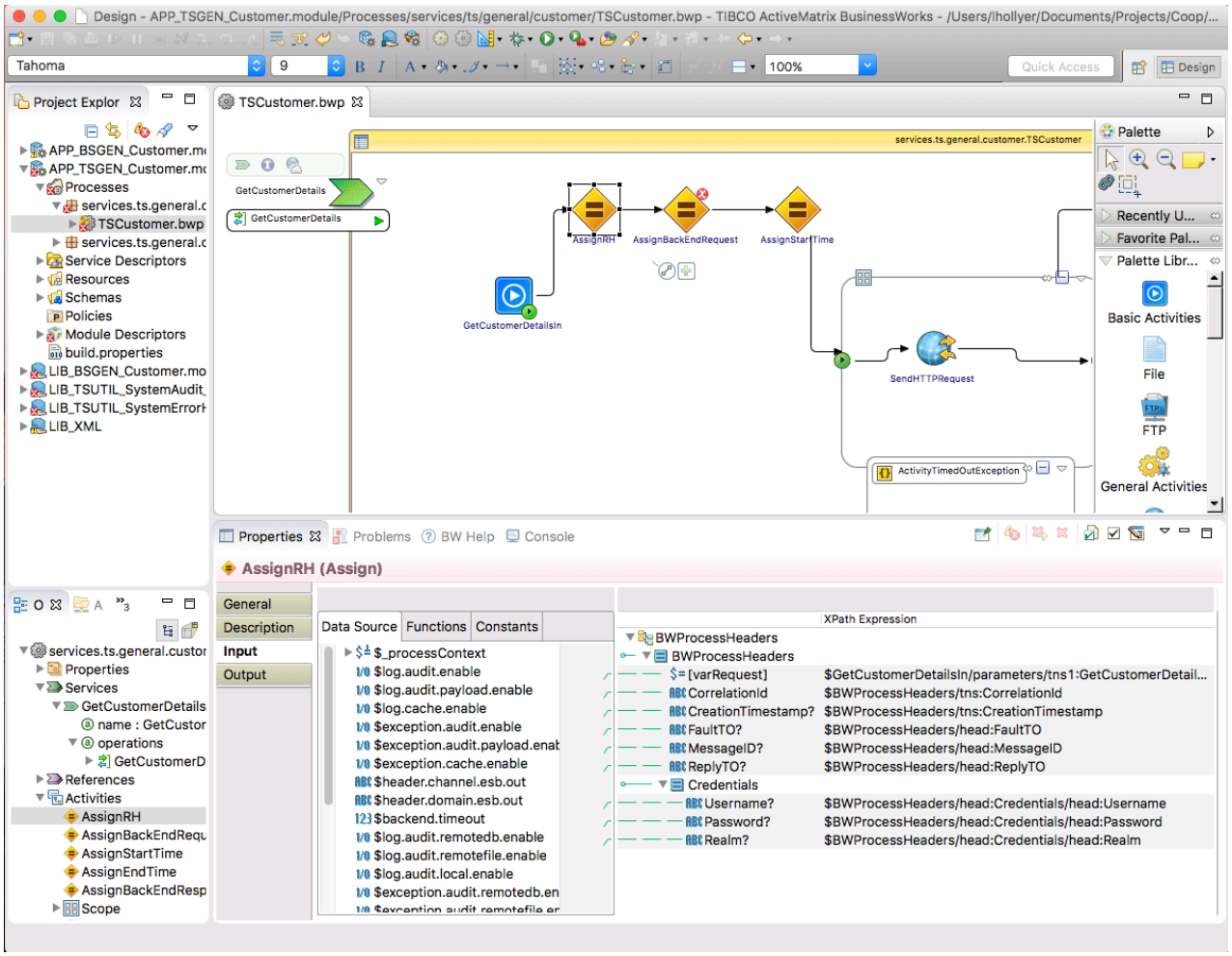


Figure 23: TS Service Provider Shared Module: AssignRH mapping

7.2.5.2 Reply_Fault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML Fault element.

Fix the configuration of the Reply_Fault task:

2. Select the service, operation and “Fault”.

Fix the mapping of the Reply_Fault task so:

2. Fault element is a copy of the CreateExceptionEvent / Response / Fault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

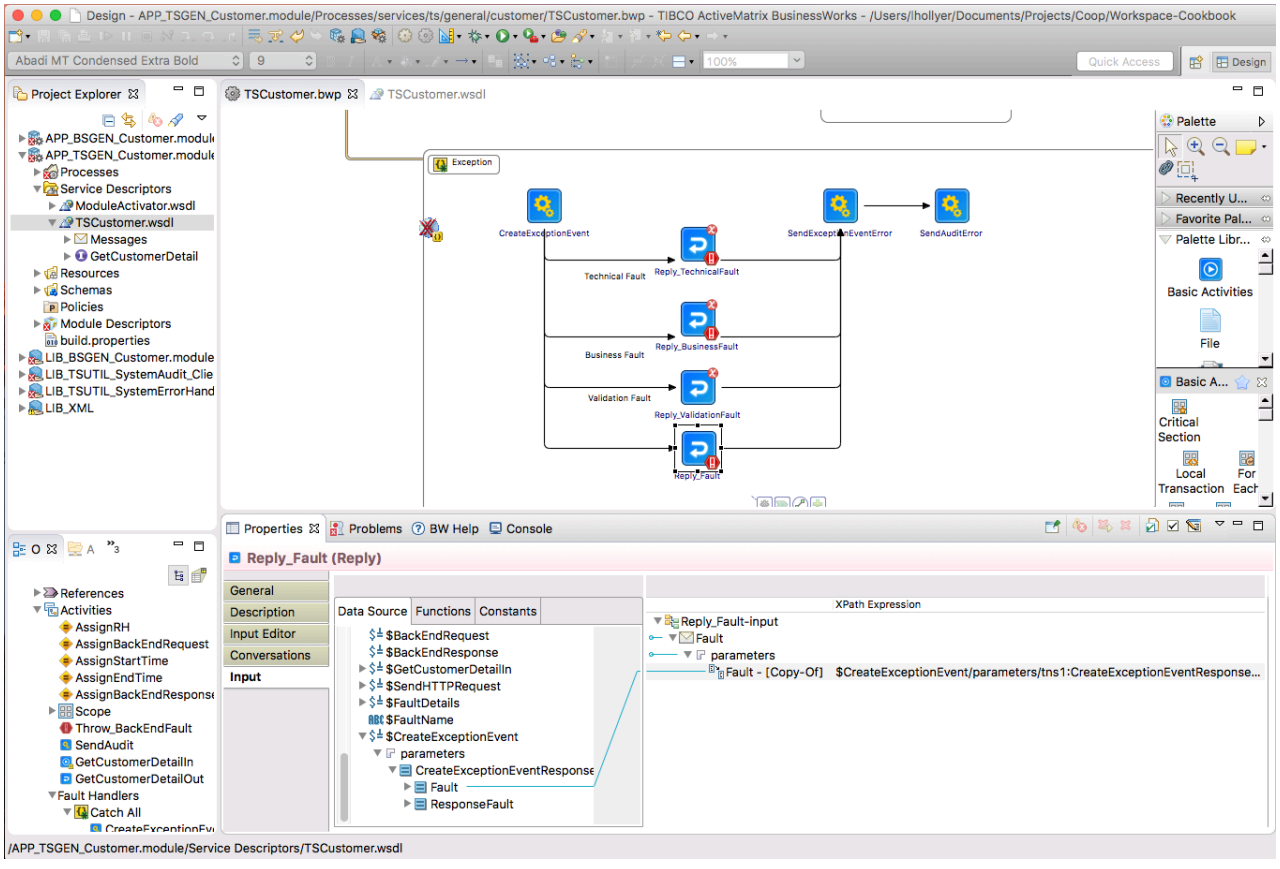


Figure 24: TS Service Provider Shared Module: Reply_Fault mapping

7.2.5.3 Reply_BusinessFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML BusinessFault element.

Fix the configuration of the BusinessFault task:

2. Select the service, operation and “BusinessFault”.

Fix the mapping of the BusinessFault task so:

2. BusinessFault element is a copy of the CreateExceptionEvent / Response / BusinessFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

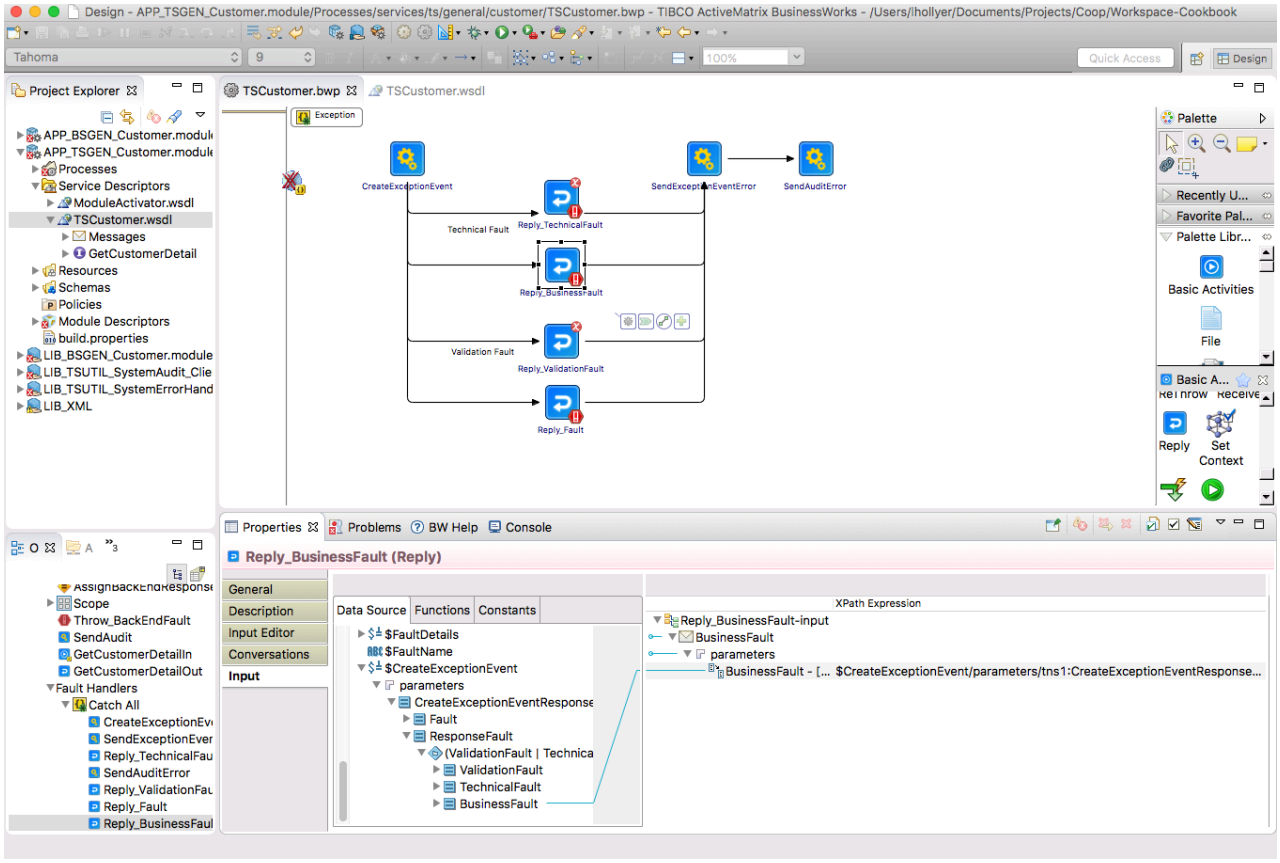


Figure 25: TS Service Provider Shared Module: Reply_BusinessFault mapping

7.2.5.4 Reply_TechnicalFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML TechnicalFault element.

Fix the configuration of the TechnicalFault task:

2. Select the service, operation and “TechnicalFault”.

Fix the mapping of the TechnicalFault task so:

2. TechnicalFault element is a copy of the CreateExceptionEvent / Response / TechnicalFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

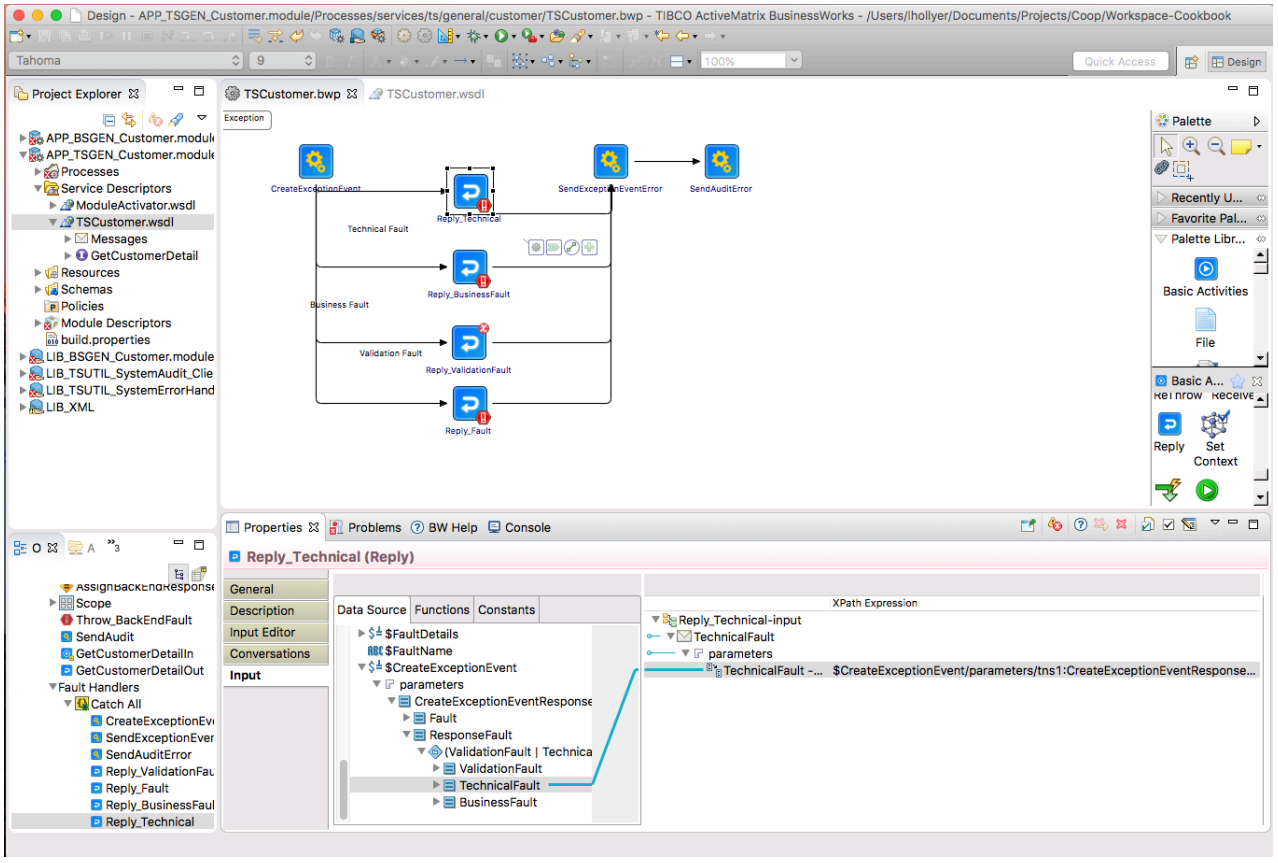


Figure 26: TS Service Provider Shared Module: Reply_TechnicalFault mapping

7.2.5.5 Reply_ValidationFault

This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML ValidationFault element.

Fix the configuration of the ValidationFault task:

2. Select the service, operation and “ValidationFault”.

Fix the mapping of the ValidationFault task so:

2. ValidationFault element is a copy of the CreateExceptionEvent / Response / ValidationFault element:

N.B. If there are no inputs listed, delete the reply and create a new reply activity and reconnect CreateExceptionEvent and SendExceptionEvent activities – make sure the line from CreateExceptionEvent’s condition is “Success with no matching condition”.

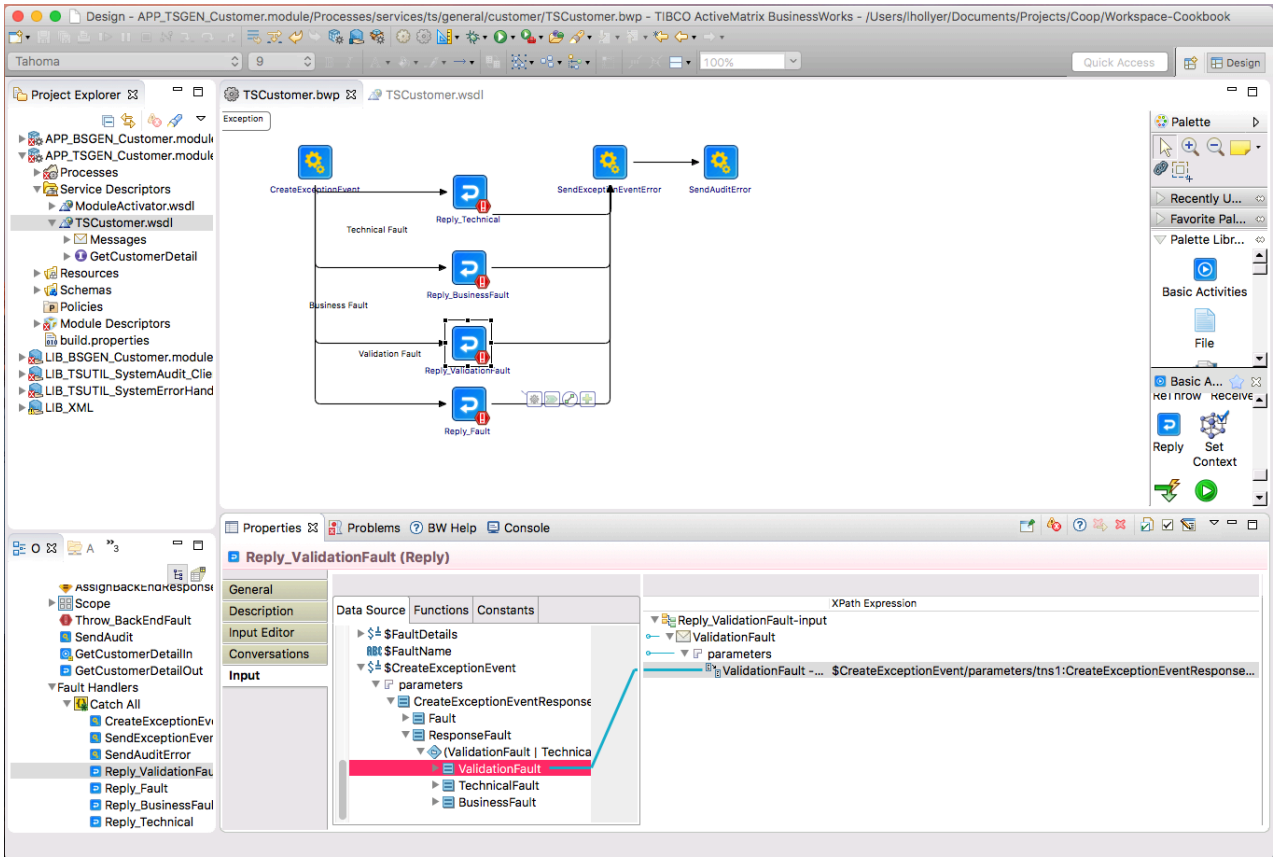


Figure 27: TS Service Provider Shared Module: Reply_ValidationFault mapping

7.2.5.6 SendExceptionEvent

This ask invokes the ESB Audit service to send one event indicating the request that was sent to the back-end and another one indicating the back-end returned a reply.

Fix the mapping of the ValidationFault task as following:

2. Add a “varBackEnd” child variable and change the value of the “varBackEnd” variable so it contains the name of the back-end system, for example “Experian”. If the back-end is .net, please use the .net service name. If the back-end is a DB you can add the name of the stored procedure as well.

7.2.5.7 SendAuditEvent_Error

This ask invokes the ESB Audit service to send one event indicating the request that was sent to the back-end, another one indicating the back-end returned a reply (if it did) and another one indicating an error happened.

Fix the mapping of the ValidationFault task as following:

2. Add a “varBackEnd” child variable and change the value of the “varBackEnd” variable so it contains the name of the back-end system, for example “Experian”. If the back-end is .net, please use the .net service name. If the back-end is a DB you can add the name of the stored procedure as well.

7.2.6 Back End on HTTP/HTTPS

If the back-end system is accessed on HTTP/HTTPS transport, you can modify the service as following.

7.2.6.1 Change the resources

4. Rename the HTTP Client resource from “services.ts.backend.servicename.Client-BACKEND-http” into “services.ts.<area>.<sevicename>.Client-<BackEndName>”.

5. Delete the JDBC resource “Client-BACKEND-jdbc”.

6. Delete the old resource package services.ts.backend.servicename.

7.2.6.2 Change the Module Properties

7. Go to the module properties.

8. Select the group “services.ts.<area>.<servicename> / <ServiceName> / backend”. In the properties pane, use the light-bulb icon to rename the group into “<BackEndName>”.

   Example:

   services.ts.<area>.<servicename> / <ServiceName> / backend / becomes
   services.ts.bline.getcardscheme / GetCardScheme / BLINE

9. In the group you renamed, change the “endpoint.uri” value to correspond to the back-end URI.

10. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”.

11. Select the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”. In the properties pane, use the light-bulb icon to rename the group into “Client-<BackEndName>”.

12. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-jdbc”.

7.2.6.3 Change Service Process (if back-end is accessed via SOAP)

2. Import the concrete WSDL into the “Service Descriptors” folder.

Change the service process as following:

Service Invocation

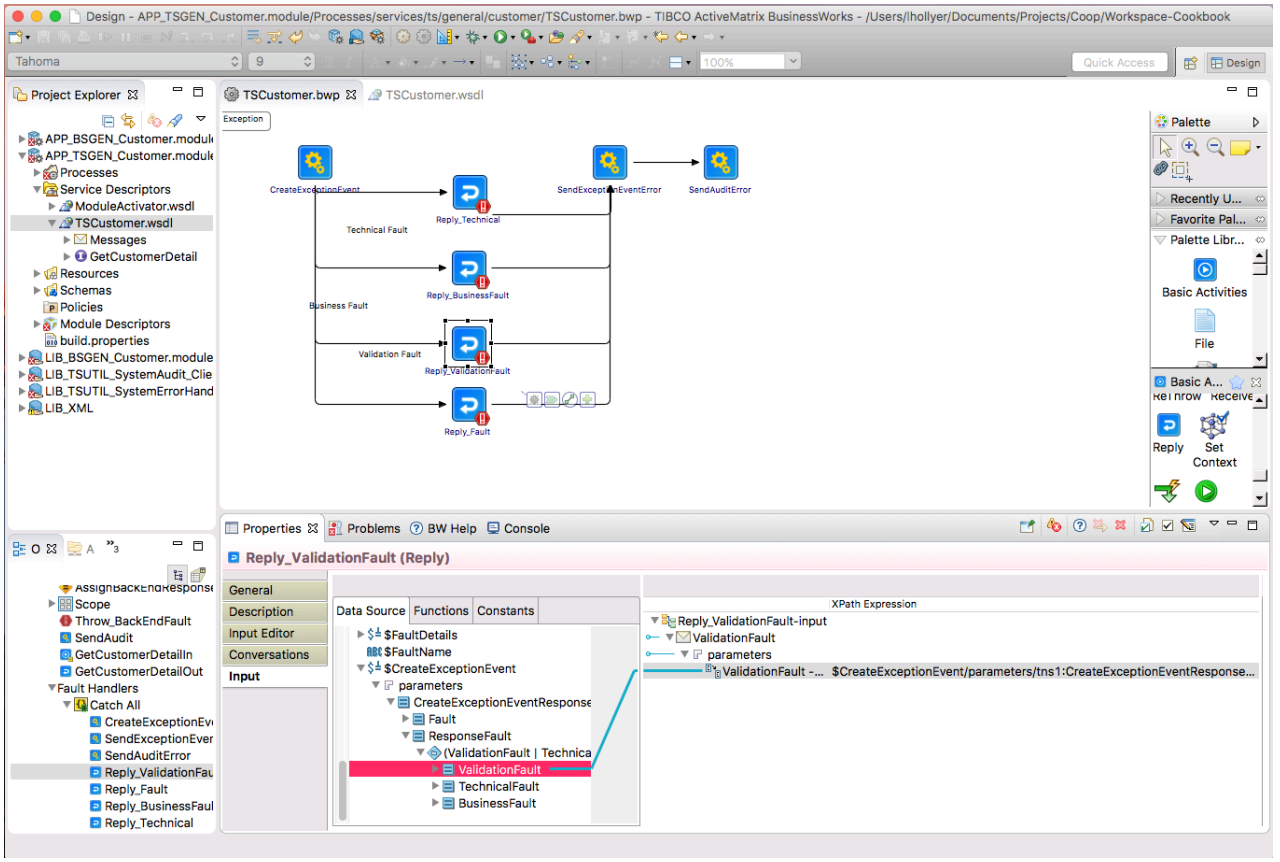
8. Remove the “SendHTTPRequest” task and replace it with an Invoke task.

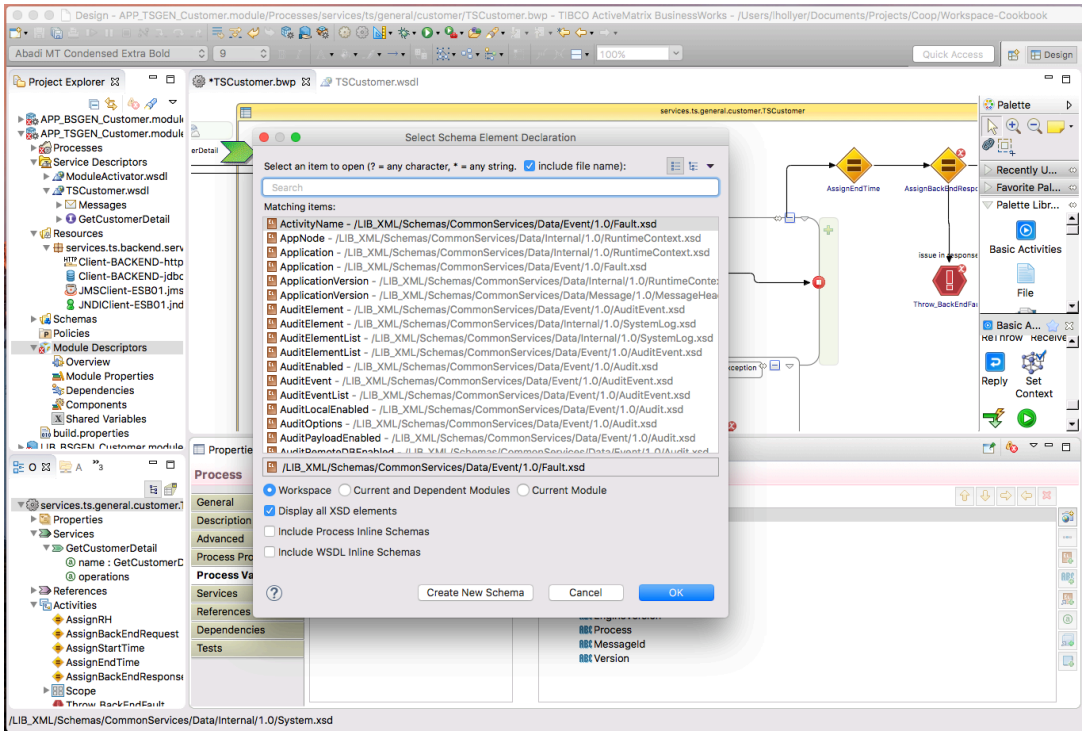
9. Configure the Service Reference and add a SOAP binding which uses:

   a. The HTTP client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.

   b. The property “services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / endpoint.uri.” for the endpoint URI.

10. Edit the process variable “$BackEndRequest” so is uses the XML schema of the back-end operation request (from its concrete WSDL).





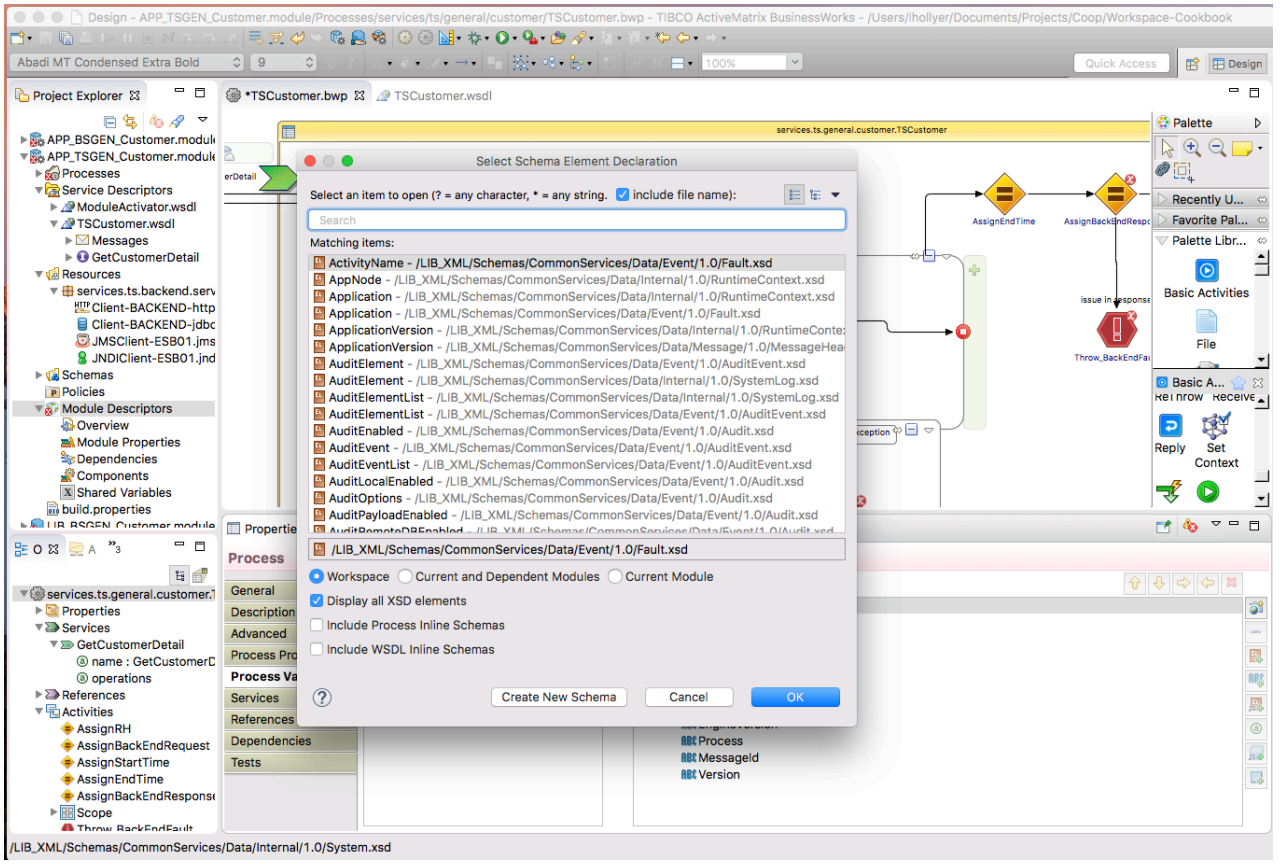
11. Edit the process variable “$BackEndResponse” so is uses the XML schema of the back-end operation response (from its concrete WSDL).
12. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).
13. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.
14. Map the response from the back-end in the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).

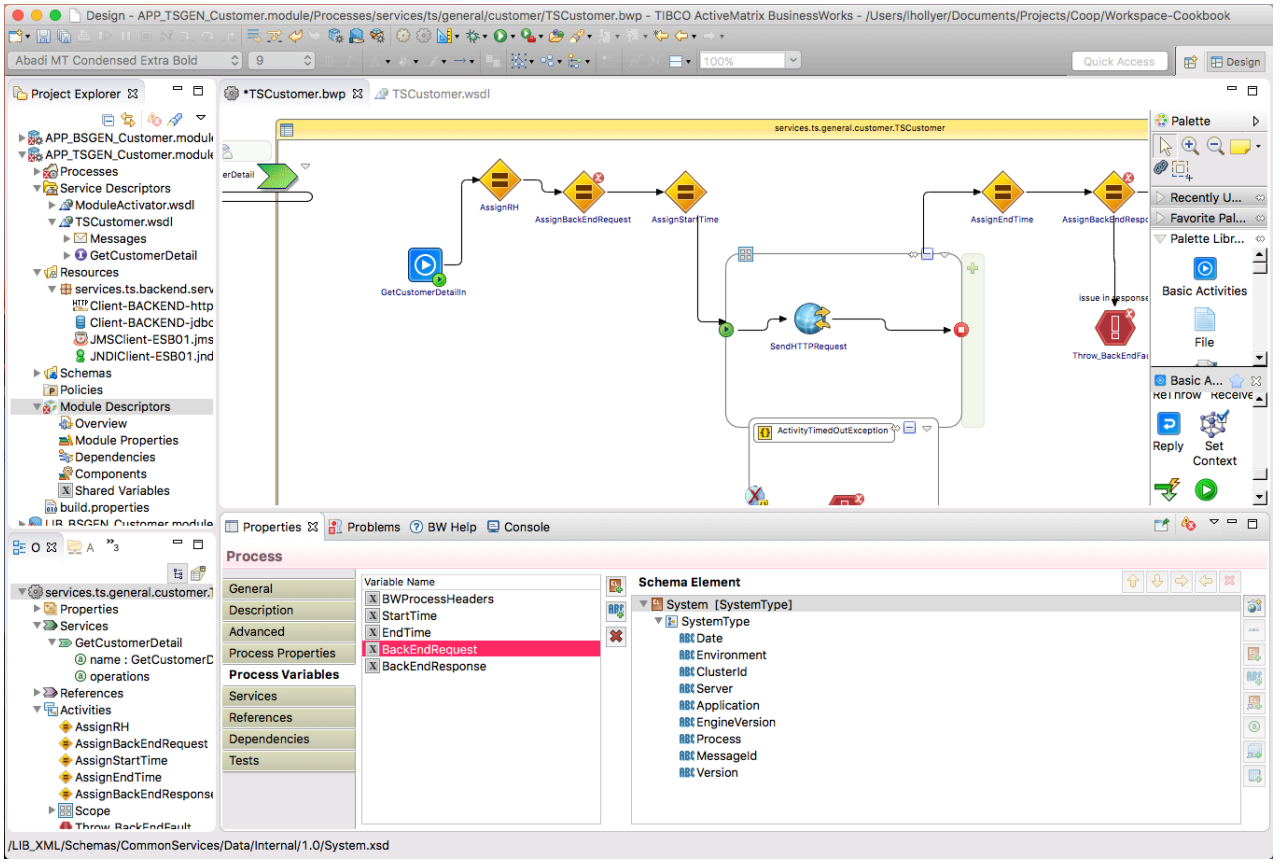
7.2.6.4 Change Service Process (if back-end is accessed via raw HTTP)

Change the service process as following:

Service Invocation

6. Edit the process variable “$BackEndRequest” so is uses the XML schema of the back-end operation request.


Select Schema Element Declaration
Select an item to open (? = any character, * = any string.
include file name):
Search
Matching items:
ActivityName - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
AppNode - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeContext.xsd
Application - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeContext.xsd
Application - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
ApplicationVersion - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/RuntimeConte
ApplicationVersion - /LIB_XML/Schemas/CommonServices/Data/Message/1.0/MessageHea
AuditElement - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditElement - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/SystemLog.xsd
AuditElementList - /LIB_XML/Schemas/CommonServices/Data/Internal/1.0/SystemLog.xsd
AuditElementList - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditEvent - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditEventList - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/AuditEvent.xsd
AuditLocalEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditOptions - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
AuditPayloadEnabled - /LIB_XML/Schemas/CommonServices/Data/Event/1.0/Audit.xsd
/LIB_XML/Schemas/CommonServices/Data/Event/1.0/Fault.xsd
Workspace
Current and Dependent Modules
Current Module
Display all XSD elements
Include Process Inline Schemas
Include WSDL Inline Schemas
Create New Schema
Cancel
OK
/LIB_XML/Schemas/CommonServices/Data/Internal/1.0/System.xsd



7. Edit the process variable “$BackEndResponse” so is uses the element “Payload” from the **LIB_XML / Schemas / CommonServices / Data / Event / 1.0 / AuditEvent.xsd** XSD.

8. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).

9. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.

10. Map the $SendHTTPRequest variable into the input the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).

7.2.6.5 Change Service Process (all cases)

Detect Back End Errors

If the service must analyze the back-end reply to detect back-end errors, this can be done in the “Throw_BusinessFault” task and the transition from “AssignBackEndResponse” task to “Throw_BusinessFault” task. In the “Throw_BusinessFault” task you can change the text and description of the issue.

If not, you can delete the “Throw_BusinessFault” task.

If the service operation can throw various SOAP faults that can be mapped to an ESB BusinessFault, you can handle them as following:

9. Right-click on the Invoke task and in the contextual menu, select the option “Catch / <fault>” where <fault> if the SOAP fault to handle.

10. In the new Catch block, add a ”Throw” task.

11. In “Input Editor” of the “Throw” task, add the XML element “ThrowableBusinessFault” from the “Schemas / CommonServices / Data / Message / Internal / Throwable” XSD.

12. Change the mapping of the task to populate the text and description of the fault.

Timeout Faults

4. In the “Throw_TimeoutFault” task, you can change the text and description of the timeout error.

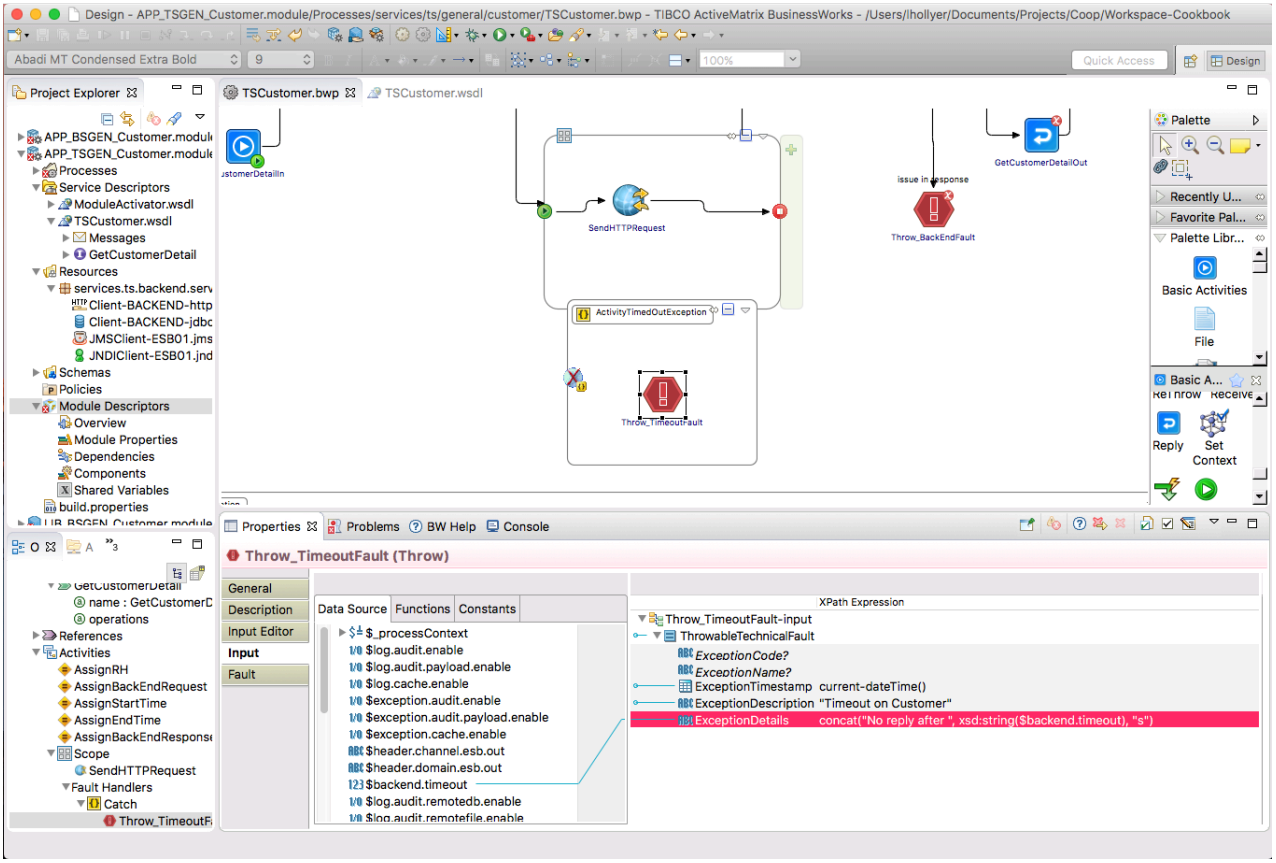


Figure 28: TS Service Provider Shared Module: Throw_TimeoutFault mapping

5. If the task has no input, fix it as following: in “Input Editor” add the XML element “ThrowableBusinessFault” from the “Schemas / CommonServices / Data / Internal / 1.0 / Internal / Throwable” XSD:

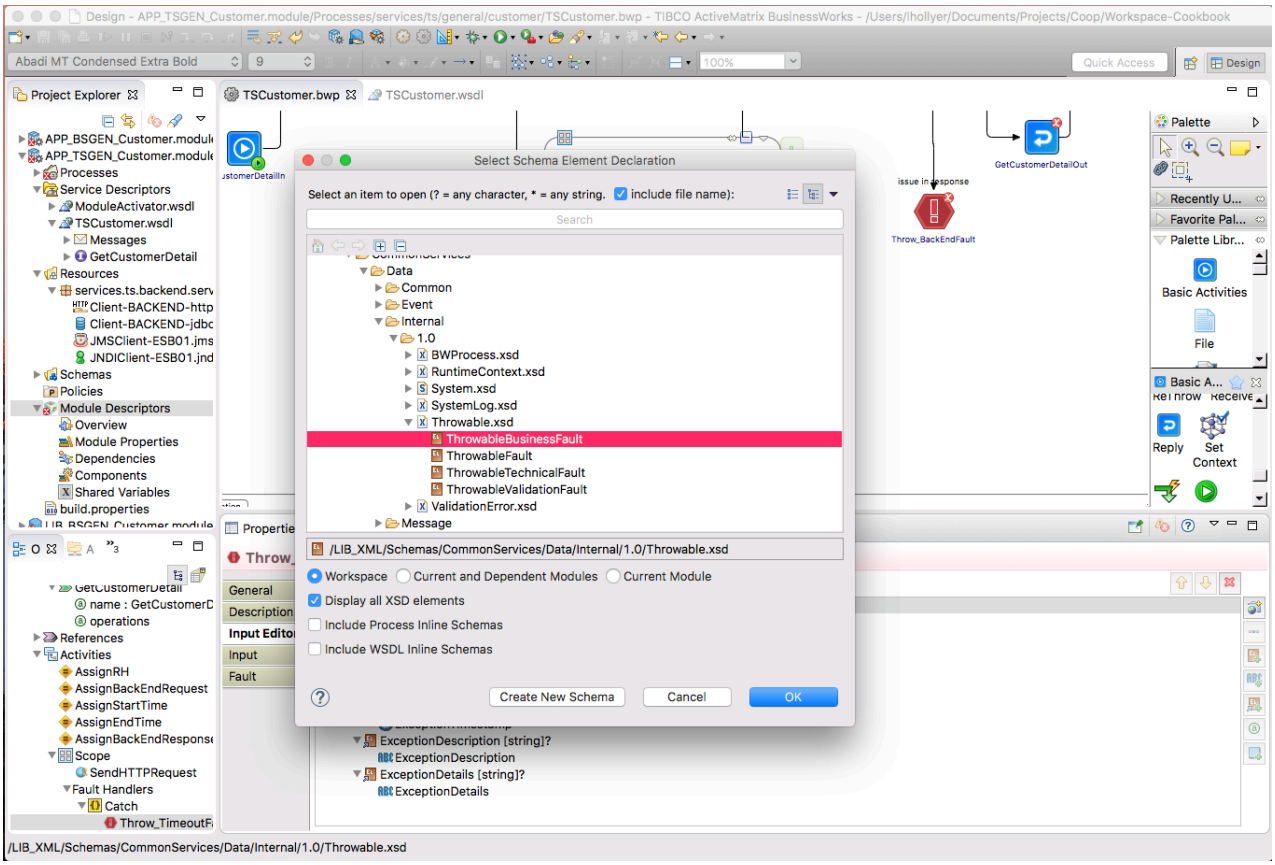


Figure 29: TS Service Provider Shared Module: Throw_TimeoutFault configuration

6. Delete and recreate the task if necessary.

7.2.6.6 Add HTTPS Resources

If the back-end is accessed via HTTPS, the following resources must be added manually and named as such:

- Resources:
  - SSL Client Configuration: services.ts.<area>.<sevicename>.Client-<BackEndName>-ssl
  - Keystore Provider Resource: services.ts.<area>.<sevicename>.Client-<BackEndName>-jks
- Module Properties
  - Keystore URL: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.url**.
  - Keystore Password: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.password**.
  - Refresh Interval: **services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / keystore.refresh.interval** (3600000 ms as default value)

7.2.7 Back End on JDBC

If the back-end system is a database, you can modify the service as following.

7.2.7.1 Change the resources

4. Rename the JDBC Client resource from “services.ts.backend.servicename.Client-BACKEND-jdbc” into “services.ts.<area>.<servicename>.Client-<BackEndName>”.

5. Delete the HTTP resource “Client-BACKEND-http”.

6. Delete the old resource package services.ts.backend.servicename.

7.2.7.2 Change the Module Properties

7. Go to the module properties.

8. Select the group “services.ts.<area>.<servicename> / <ServiceName> / backend / BACKEND”. In the properties pane, use the light-bulb icon to rename the group into “<BackEndName>”.

9. In the group you renamed, delete the “endpoint.uri” property.

10. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”.

11. Select the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-jdbc”. In the properties pane, use the light-bulb icon to rename the group into “Client-<BackEndName>”.

12. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”.

7.2.7.3 Change Service Process

Change the service process as following:

7. Remove the “SendHTTPRequest” task and replace it with a JDBC task.

8. Configure the JDBC tasks so it uses:

    a. The JDBC client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.

    b. The property “services.ts.<area>.<servicename> / <ServiceName> / backend / <BackEndName> / timeout” for the timeout.

9. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).

10. Map the response from the back-end in the “AssignBackEndResponse” task. This task stores the response into the process variable $BackEndResponse so it can be re-used in other tasks (for the SystemAudit tasks for example).

11. If the service must analyze the back-end reply to detect back-end errors, this can be done in the “Throw_BusinessFault” task and the transition from “AssignBackEndResponse” task to “Throw_BusinessFault” task. In the “Throw_BusinessFault” task you can change the text and description of the issue.

    If not, you can delete the “Throw_BusinessFault” task.

12. In the “Throw_TimeoutFault” task, you can change the text and description of the timeout error.

7.3 Implement the Service

7.3.1 Service Response

In the <Operation>Out task, you MUST map the ResponseHeader element as following:

- **CorrelationId**: $BWProcessHeaders / CorrelationId
- **MessageID**: $BWProcessHeaders / MessageID
- **Version**: $BWProcessHeaders / Response / Version
- **ResponseDateTime**: current-dateTime()

N.B. if <Operation>Out does not have the ResponseHeader element, check the WSDL location imports the Schema in **APP**_TS<Service>.module’s XSD, and check the XSD has the ResponseHeader set to ResponseHeaderType from LIB_XML.

7.3.2 Invoking another ESB service

If you invoke another ESB service (BS or TS), you MUST map the RequestHeader element as following:

- **CorrelationId**: $BWProcessHeaders / CorrelationId
- **MessageID**: $BWProcessHeaders / MessageID
- **Version**: The version of the invoked service, for example “1.0”.
- **RequestDateTime**: current-dateTime()

N.B. if <Operation>Out does not have the ResponseHeader element, check the WSDL location imports the Schema in APP_TS<Service>.module’s XSD, and check the XSD has the RequestHeader set to RequestHeaderType from LIB_XML.

7.3.3 Validate the Request

Some service may require additional input data validation (besides what is defined inside the service schema, which BW validates by default). In such case, you must implement the validation as following:

- The steps are the same as for a business service (see section 4.3.5), but the “Throw_ValidationFault” happens between the “AssignRH” task and “AssignBackEndRequest” task.

7.3.4 Module Properties, Processes, Service Invocation

The rest of the procedure is the same as for a technical service provider in a Shared Module, please refer to the previous chapter for all required procedure information, except for the next sections, which are specific to an application.

7.3.5 Change the JMS Queue name

Edit the module properties as following:

1. Change the value of the property **services.ts.<area>.<servicename> / <ServiceName> / backend / BACKEND / jms.queue** into Coop.Kenya.Q.RQ.TS.<Area>.<ServiceName>.SOAP11.1.

7.3.6 Change the JMS Connection

1. Rename the JNDI Client resource from “services.ts.backend.servicename.JNDIClient-ESB01” into “servies.ts.<area>.<sevicename>.JNDIClient-ESB01” (only the package changes).

2. Rename the JNDI Client resource from “services.ts.backend.servicename.JMSClient-ESB01” into “servies.ts.<area>.<sevicename>.JMSClient-ESB01” (only the package changes).

3. Delete the old resource package services.ts.backend.servicename.

7.3.7 Create the SOAP Binding

Once your service process has been renamed and moved to the proper Process Package, you must create its SOAP over JMS binding.

1. Expand “Module Properties” in the module Project Explorer.

2. Double-click on “Module Descriptors / Components”:

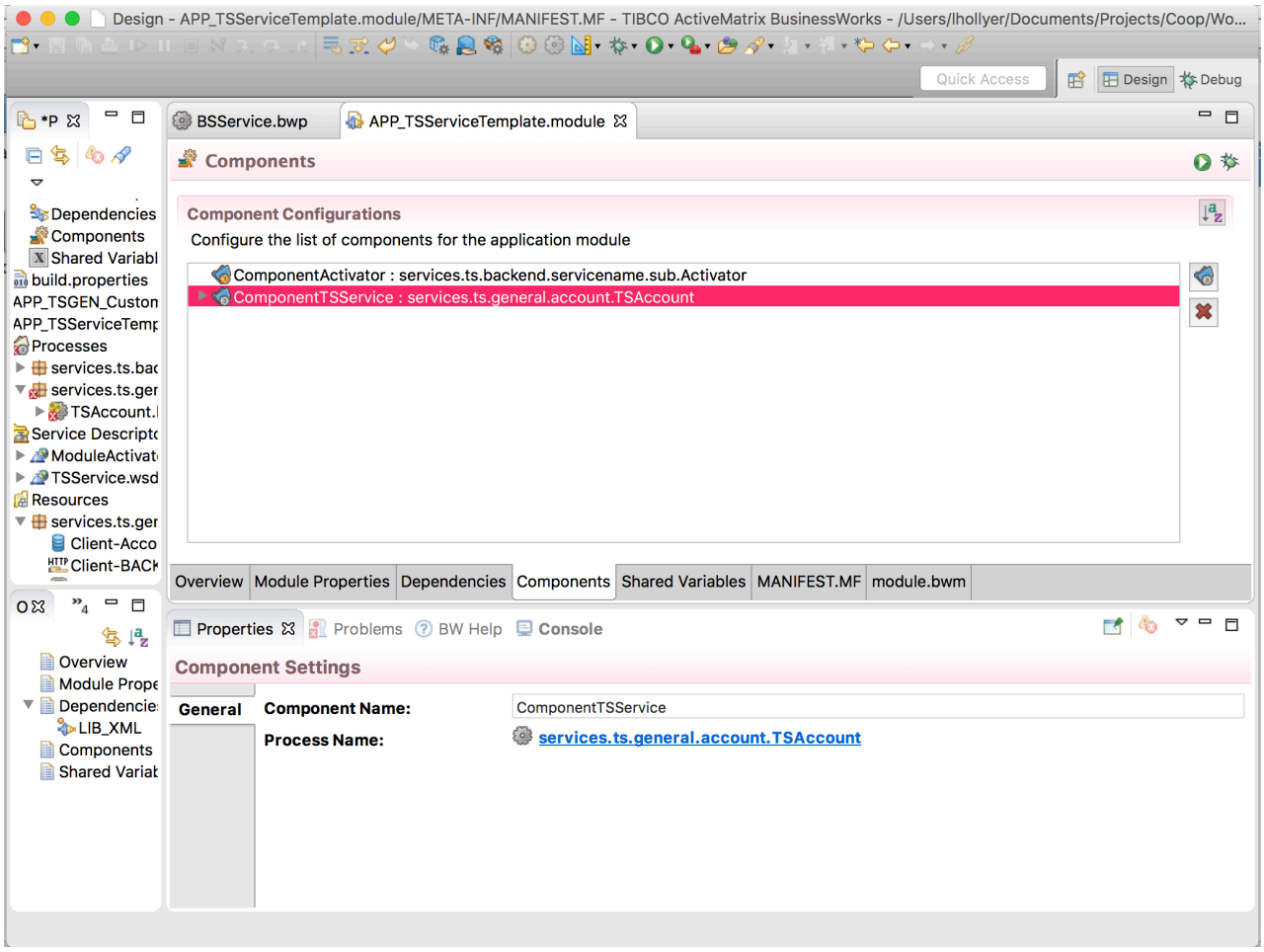


Figure 30: Create the SOAP over JMS Interface: Module Components

3. Click on the arrow on the left of “Component<serviceName>”, select “<serviceName>” and select “Properties” in the pane below (you may to click several times on <serviceName> to see the properties appear as below):

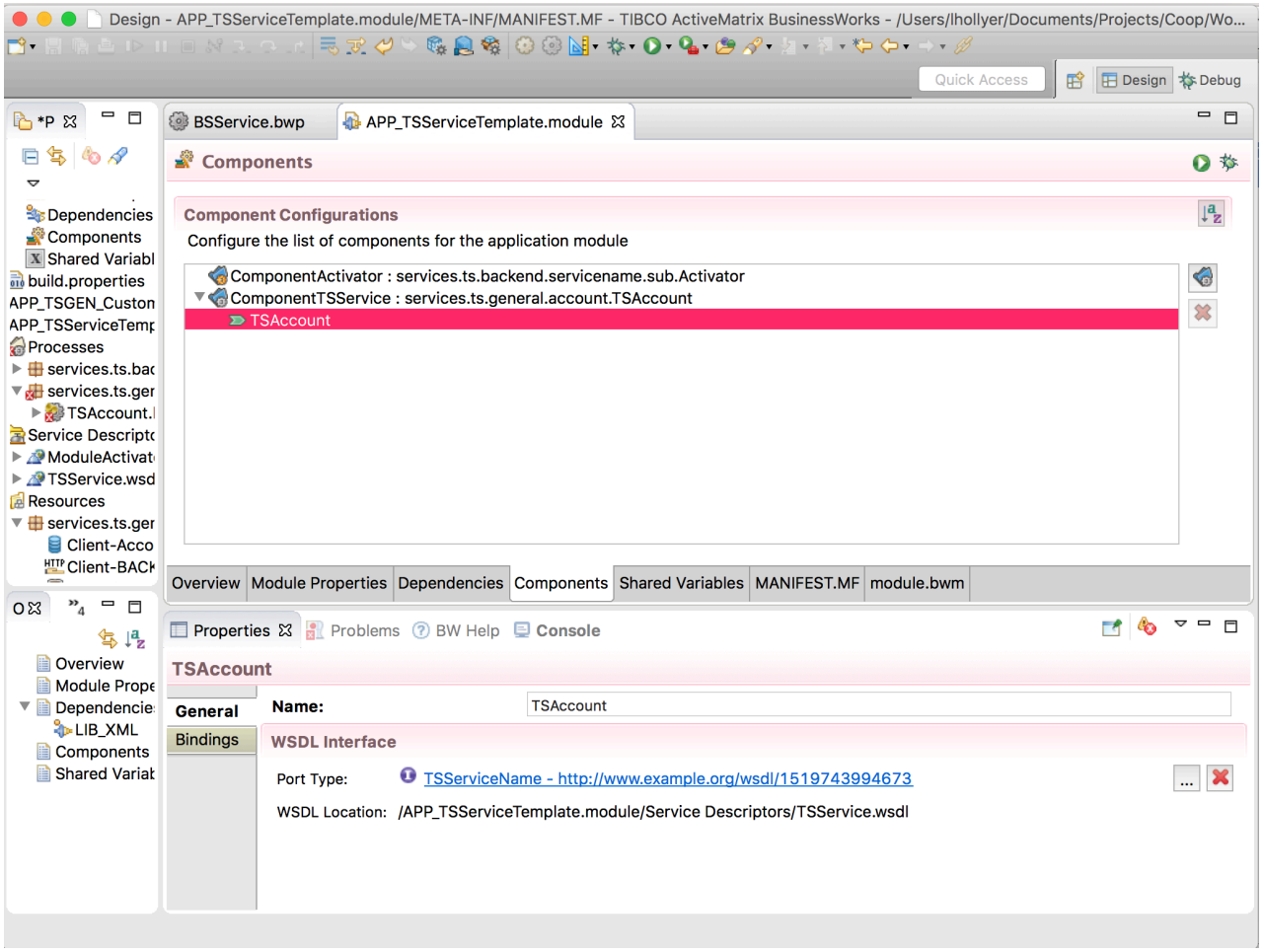


Figure 31: Create the SOAP over JMS Interface: Service Settings

4. Select “Bindings” and click on the “+” button.
5. In the “Add Binding” dialog box:
    a. Select “SOAP Binding”,
    b. Check the box “Select / Create Required Resources”.

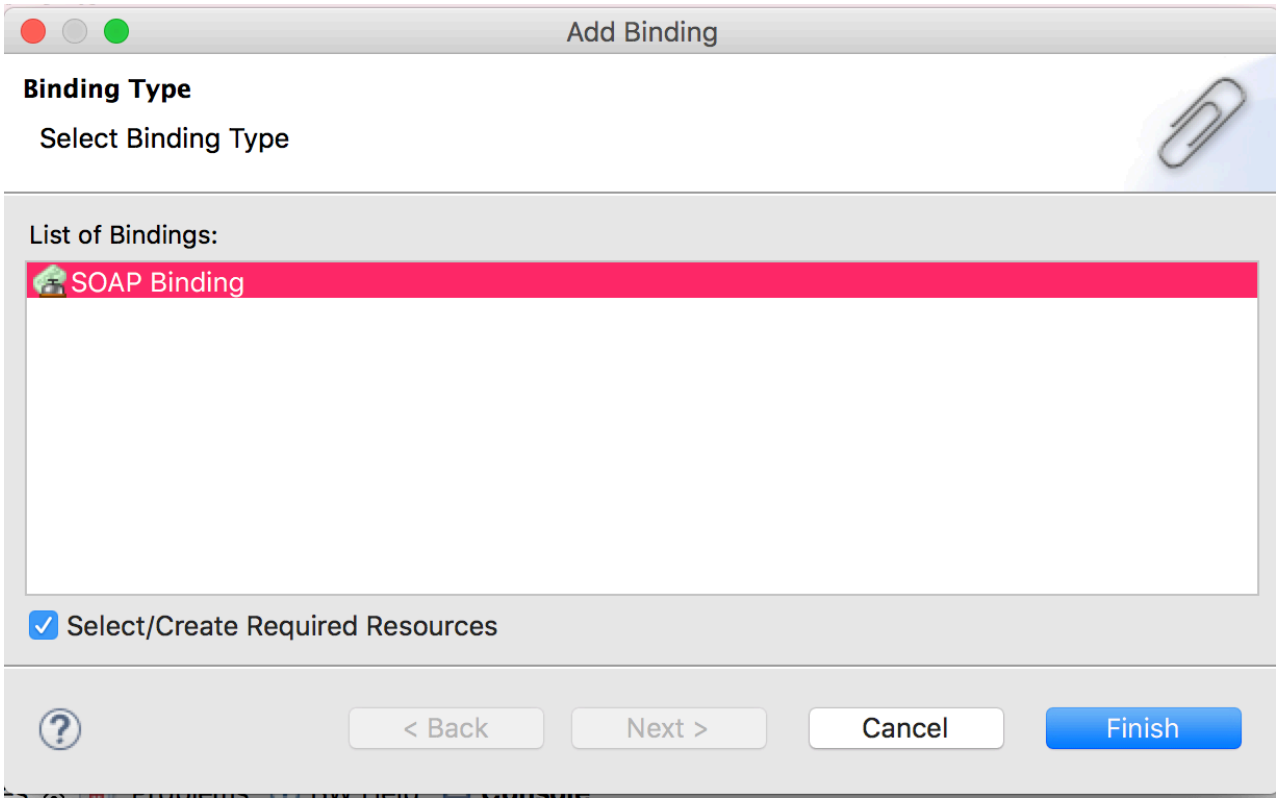


Figure 32: Create the SOAP over JMS Interface: New SOAP Binding

6. Click on the “Finish” button.

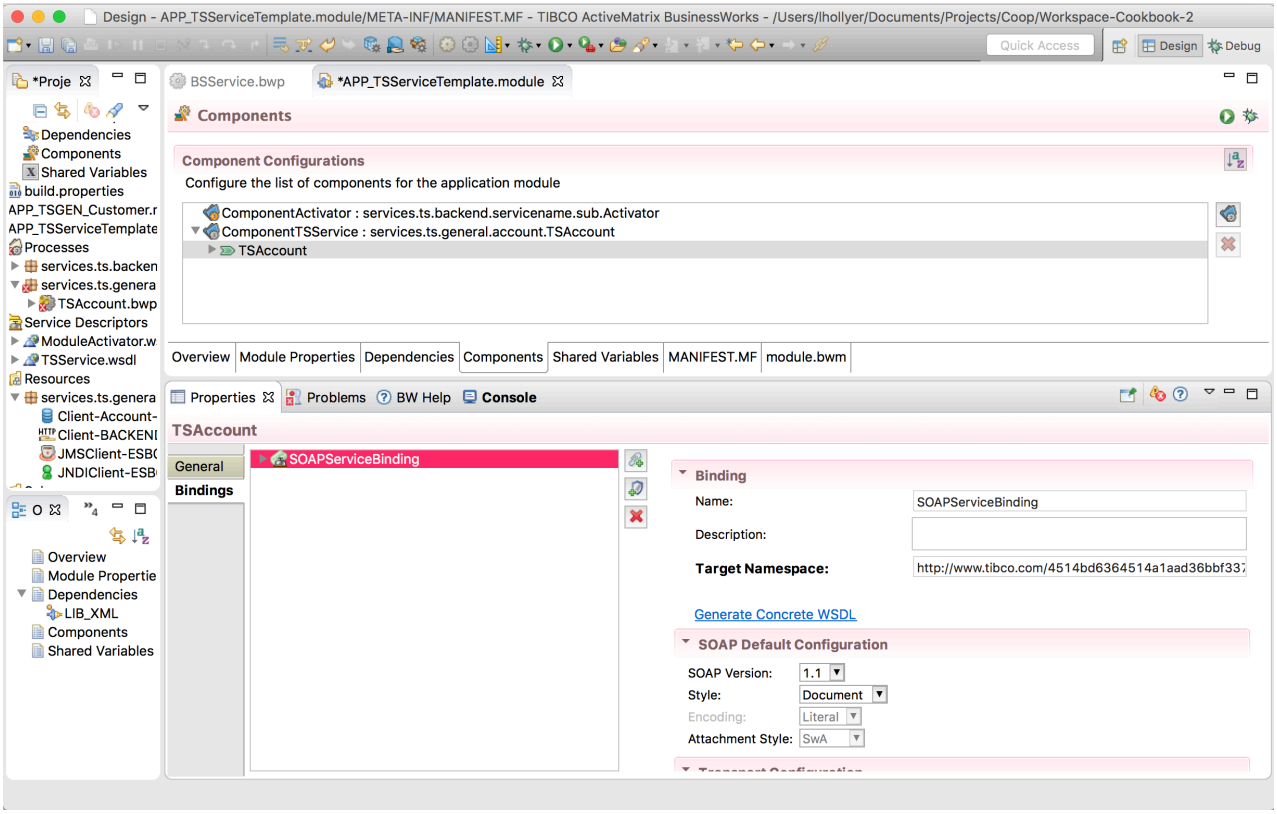


Figure 33: Create the SOAP over JMS Interface: New SOAP Binding

7. In the new “Binding” panel that appeared:

    a. Select “JMS” in the “Transport Type”.

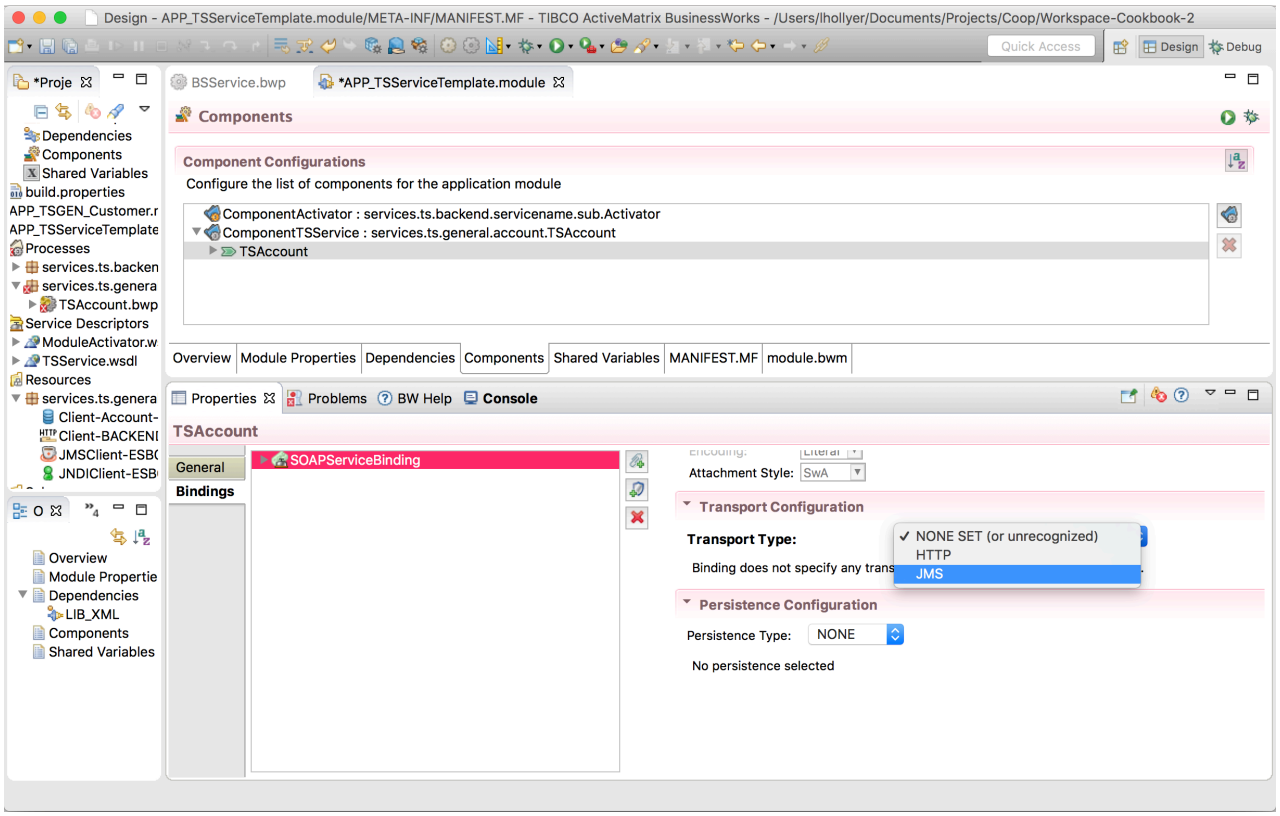


Figure 34: Create the SOAP over JMS Interface: Transport Type

8. In the “Select Jms… Dialog” box, select the JMS connection from the current module (the JMS Client resource renamed in previous steps)/

9. Now, back into the “Binding” panel:

    b. Select Acknowledgment Mode as “Auto”,

    c. Select JMS Message Type” as “Text”,

    d. Select Messaging Style as “Queue”.

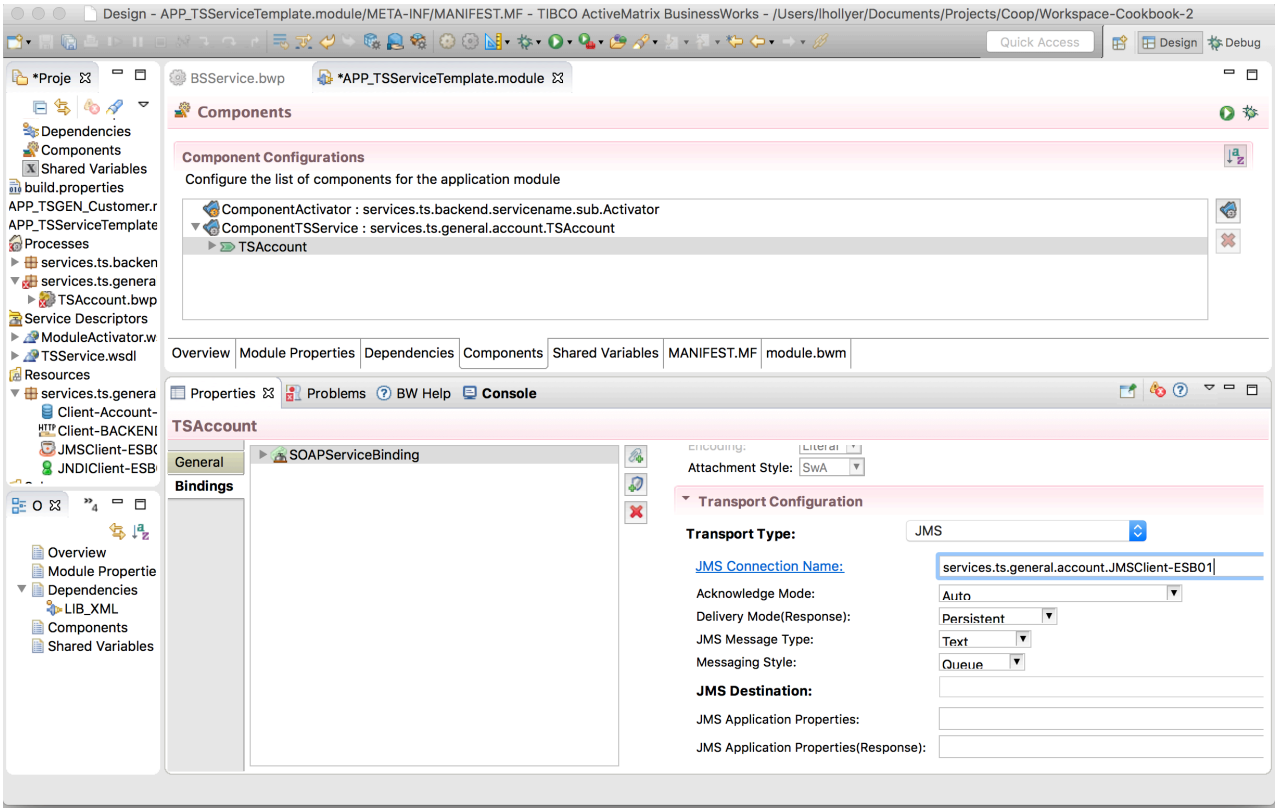


Figure 35: Create the SOAP over JMS Interface: JMS Settings

10. Change the JMS Destination so it uses the module property services.ts.<area>.<servicename> / <ServiceName> / jms.queue:

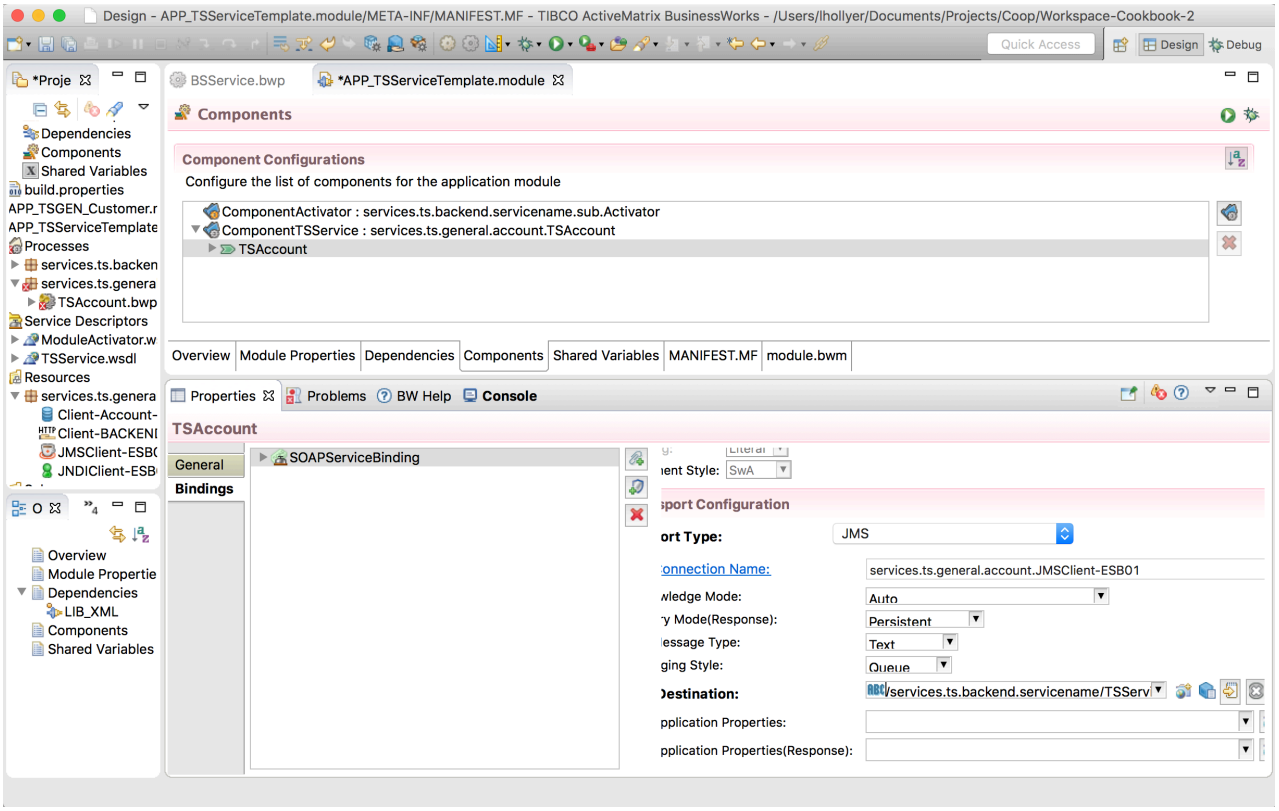
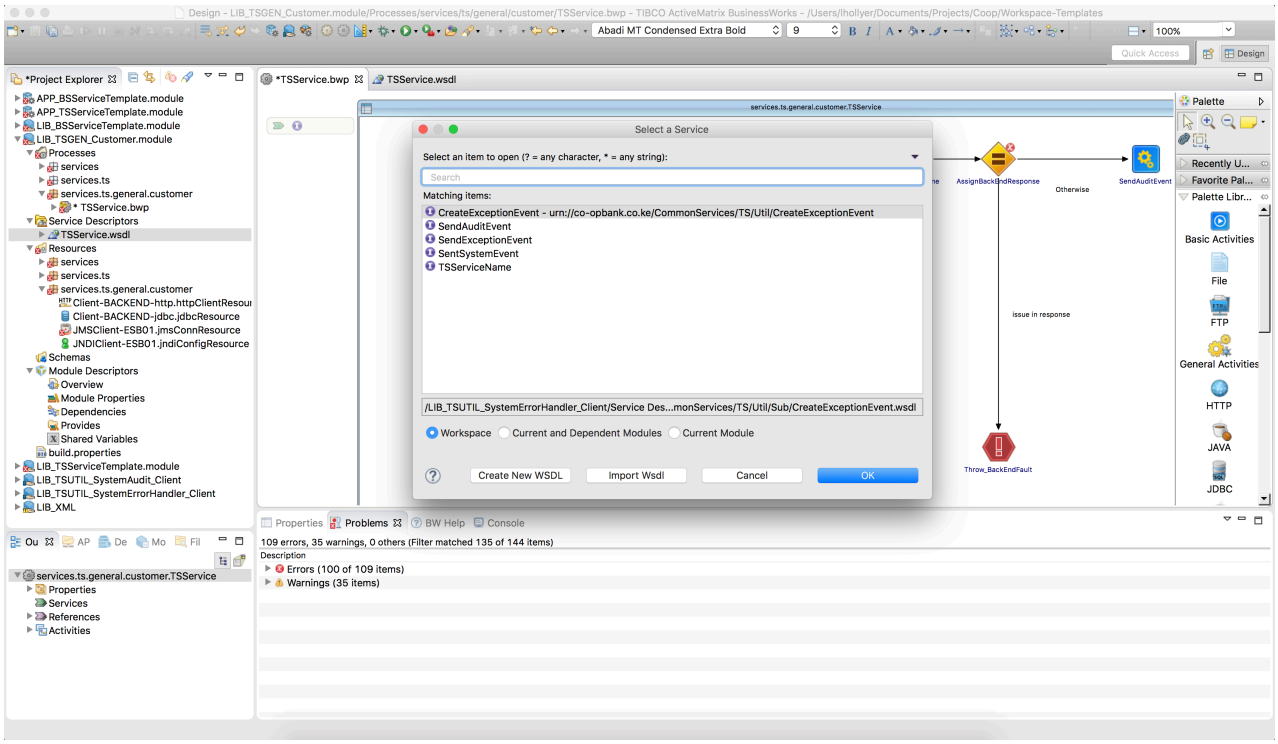
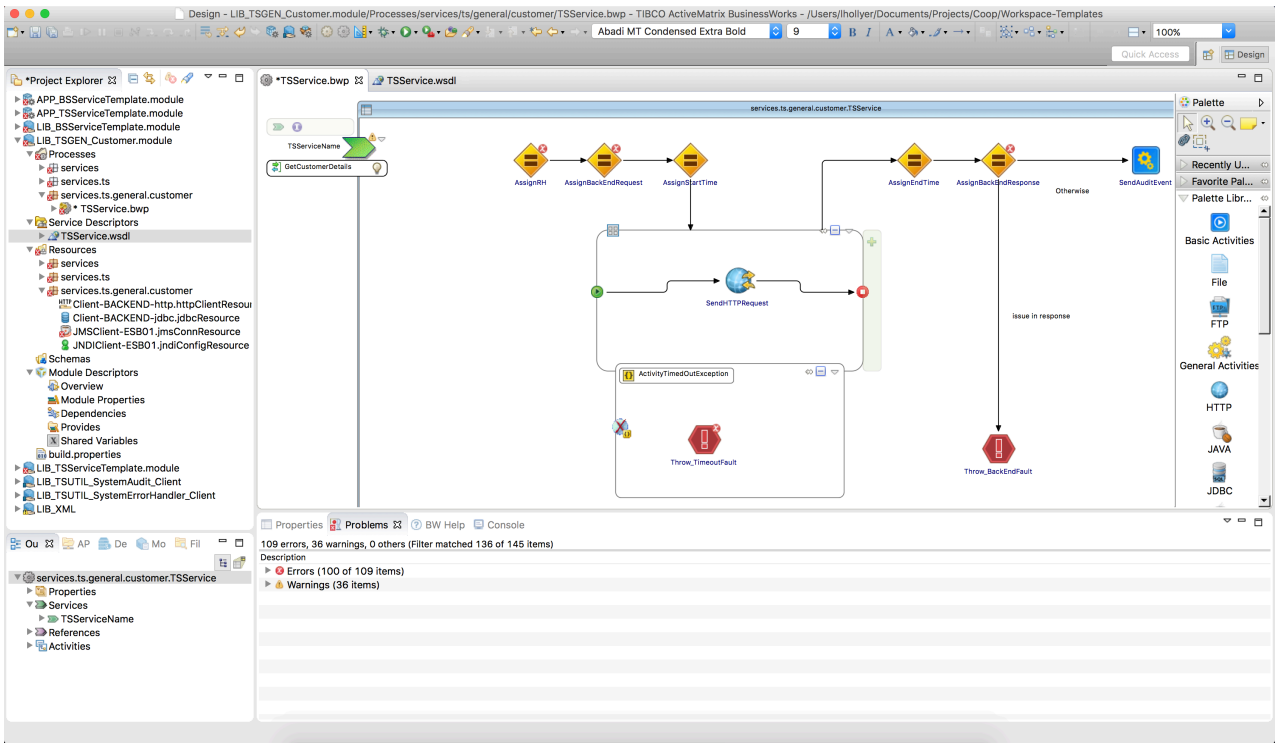


Figure 36: Create the SOAP over JMS Interface: Queue Name as Property

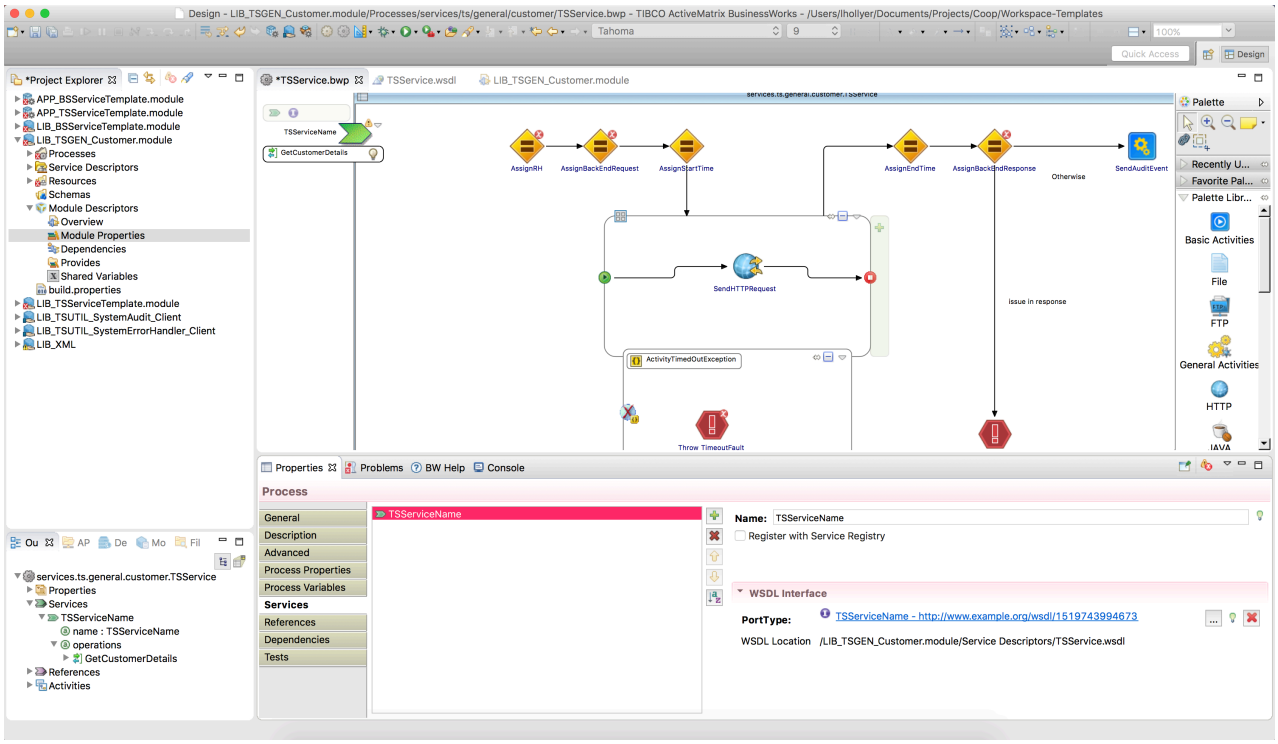
11. Save.



1. Select TS<ServiceName> and click OK.



2. Go to the process's properties and select the service on the Services tab.



7.3.8 Create the EMS deployment script

You must create one script that can be run against the EMS Server with tibemsadmin tool to deploy the queue required by this service.

1. In the <GIT> / trunk / EMS folder create the text file “deploy-TS<AR>_<ServiceName>.ems” with the following content:

```
#################################################################################
# This script deploys EMS resource for the service TS<AR>_<ServiceName>
#
# Version 1.0.0 - <author> - <date>
#
# Changes:
# 1.0.0 - <date> - First version
#
#################################################################################

#################################################################################
# Queues

create queue <queue>
secure,sender_name_enforced,maxmsgs=100000,maxbytes=1GB,expiration=10,overflowPol
icy=discardOld
grant queue <queue> user=bw send,receive

#################################################################################

commit

#################################################################################
###  END OF FILE  ###############################################################
#################################################################################
```

2. Replace <author>, <date>, <queue> by the appropriate values.

7.3.9 Create the EMS undeployment scripts

You must create one script that can be run against the EMS Server with tibemsadmin tool to undeploy the queue required by this service.

1. In the **<GIT> / trunk / EMS / rollback** folder create the text file “undeploy-TS<AR>_<ServiceName>.ems” with the following content:

```
#################################################################################
This script un-deploys EMS resource for the service TS<AR>_<ServiceName>
#
# Version 1.0.0 - <author> - <date>
#
# Changes:
# 1.0.0 - <date> - First version
#
#################################################################################

#################################################################################
# Queues

delete queue <queue>

#################################################################################

commit

#################################################################################
###  END OF FILE  ###############################################################
#################################################################################
```

2. Replace <author>, <date>, <queue> by the appropriate values.