TESCO Bank

ESB

How To: Create BW Services from Templates

Document Version: v1.0.0

Document Date: May 2017

Copyright Notice

COPYRIGHT© 2010 TIBCO Software Inc. This document is unpublished and the foregoing notice is affixed to protect TIBCO Software Inc. in the event of inadvertent publication. All rights reserved. No part of this document may be reproduced in any form, including photocopying or transmission electronically to any computer, without prior written consent of TIBCO Software Inc. The information contained in this document is confidential and proprietary to TIBCO Software Inc. and may not be used or disclosed except as expressly authorized in writing by TIBCO Software Inc. Copyright protection includes material generated from our software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks

Technologies described herein are either covered by existing patents or patent applications are in progress. All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged.

Confidentiality

The information in this document is subject to change without notice. This document contains information that is confidential and proprietary to TIBCO Software Inc. and may not be copied, published, or disclosed to others, or used for any purposes other than review, without written authorization of an officer of TIBCO Software Inc. Submission of this document does not represent a commitment to implement any portion of this specification in the products of the submitters.

Content Warranty

The information in this document is subject to change without notice. THIS DOCUMENT IS PROVIDED “AS IS” AND TIBCO MAKES NO WARRANTY, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TIBCO Software Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

For more information, please contact:

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

**Table of Contents**

[1 Document Control 6](#_Toc482786066)

[1.1 Versioning Information 6](#_Toc482786067)

[1.2 Distribution Information 6](#_Toc482786068)

[1.2.1 TESCO Bank 6](#_Toc482786069)

[1.2.2 TIBCO 6](#_Toc482786070)

[1.3 Reference Documents 6](#_Toc482786071)

[1.4 Document Acceptance 7](#_Toc482786072)

[2 Introduction 8](#_Toc482786073)

[2.1 Scope 8](#_Toc482786074)

[2.2 Purpose 8](#_Toc482786075)

[2.3 Audience 8](#_Toc482786076)

[2.4 Acronyms 8](#_Toc482786077)

[2.5 Issues 9](#_Toc482786078)

[2.6 Risks 9](#_Toc482786079)

[3 Create a Business Service Provider as an Application 10](#_Toc482786080)

[3.1 Template Overview 10](#_Toc482786081)

[3.2 Pre-Requisites 10](#_Toc482786082)

[3.2.1 Create the Application Module 10](#_Toc482786083)

[3.2.2 Change the Module Properties 13](#_Toc482786084)

[3.2.3 Rename the Resources 14](#_Toc482786085)

[3.2.4 Create the Process Packages 14](#_Toc482786086)

[3.2.5 Change the Service Process 15](#_Toc482786087)

[3.2.6 Fix the Process Mappings 20](#_Toc482786088)

[3.2.7 Create the SOAP Binding 26](#_Toc482786089)

[3.3 Implement the Service 26](#_Toc482786090)

[3.3.1 Audit Fields 26](#_Toc482786091)

[3.3.2 Business Logic 26](#_Toc482786092)

[3.3.3 AssignResponse 26](#_Toc482786093)

[3.3.4 Error Handling 27](#_Toc482786094)

[3.3.5 Invoke another ESB service 28](#_Toc482786095)

[3.3.6 Validate the Request 29](#_Toc482786096)

[3.4 Finalization 30](#_Toc482786097)

[4 Create a Business Service Provider as a Shared Module 31](#_Toc482786098)

[4.1 Template Overview 31](#_Toc482786099)

[4.2 Pre-Requisites 31](#_Toc482786100)

[4.3 Procedure 31](#_Toc482786101)

[4.3.1 Create the Application Module 31](#_Toc482786102)

[4.3.2 Module Properties, Processes, Service Invocation 33](#_Toc482786103)

[4.4 Finalization 34](#_Toc482786104)

[5 Create a Technical Service Provider as a Shared Module 35](#_Toc482786105)

[5.1 Template Overview 35](#_Toc482786106)

[5.2 Pre-Requisites 35](#_Toc482786107)

[5.3 Procedure 35](#_Toc482786108)

[5.3.1 Create the Application Module 35](#_Toc482786109)

[5.3.2 Change the Module Properties 38](#_Toc482786110)

[5.3.3 Create the Process Packages 39](#_Toc482786111)

[5.3.4 Change the Service Process 40](#_Toc482786112)

[5.3.5 Fix the Process Mappings 44](#_Toc482786113)

[5.3.6 Back End on HTTP/HTTPS 51](#_Toc482786114)

[5.3.7 Back End on JDBC 59](#_Toc482786115)

[5.4 Implement the Service 61](#_Toc482786116)

[5.4.1 Service Response 61](#_Toc482786117)

[5.4.2 Invoking another ESB service 61](#_Toc482786118)

[5.4.3 Validate the Request 61](#_Toc482786119)

[6 Create a Technical Service Provider as an Application 62](#_Toc482786120)

[6.1 Template Overview 62](#_Toc482786121)

[6.2 Pre-Requisites 63](#_Toc482786122)

[6.2.1 Create the Application Module 63](#_Toc482786123)

[6.2.2 Module Properties, Processes, Service Invocation 63](#_Toc482786124)

[6.2.3 Change the JMS Queue name 63](#_Toc482786125)

[6.2.4 Change the JMS Connection 64](#_Toc482786126)

[6.2.5 Create the SOAP Binding 65](#_Toc482786127)

[6.2.6 Create the EMS deployment script 70](#_Toc482786128)

[6.2.7 Create the EMS undeployment scripts 71](#_Toc482786129)

[Appendix 72](#_Toc482786130)

[6.3 How to Upgrade to Version 0.4.4 72](#_Toc482786131)

[6.3.1 General 72](#_Toc482786132)

[6.3.2 Business Services 72](#_Toc482786133)

[6.3.3 Technical Services 73](#_Toc482786134)

**Table of Figures**

[Figure 1: Throw\_ValidationFault sample 29](#_Toc482786135)

[Figure 2: TS Service Provider Shared Module: Module Name 37](#_Toc482786136)

[Figure 3: TS Service Provider Shared Module: Module Properties 38](#_Toc482786137)

[Figure 4: TS Service Provider Shared Module: Resource Properties 39](#_Toc482786138)

[Figure 5: TS Service Provider Shared Module: Process Name 40](#_Toc482786139)

[Figure 6: TS Service Provider Shared Module: Implement Service Operation 41](#_Toc482786140)

[Figure 7: TS Service Provider Shared Module: Change input transition 42](#_Toc482786141)

[Figure 8: TS Service Provider Shared Module: Reply Transition 43](#_Toc482786142)

[Figure 9: TS Service Provider Shared Module: AssignRH mapping 44](#_Toc482786143)

[Figure 10: TS Service Provider Shared Module: Reply\_Fault mapping 45](#_Toc482786144)

[Figure 11: TS Service Provider Shared Module: Reply\_TimeoutFault mapping 46](#_Toc482786145)

[Figure 12: TS Service Provider Shared Module: Reply\_BackEndFault mapping 47](#_Toc482786146)

[Figure 13: TS Service Provider Shared Module: Reply\_ValidationFault mapping 48](#_Toc482786147)

[Figure 14: TS Service Provider Shared Module: Reply\_AuthenticationFault mapping 49](#_Toc482786148)

[Figure 15: TS Service Provider Shared Module: Throw\_TimeoutFault mapping 56](#_Toc482786149)

[Figure 16: TS Service Provider Shared Module: Throw\_TimeoutFault configuration 57](#_Toc482786150)

[Figure 17: Create the SOAP over JMS Interface: Module Components 65](#_Toc482786151)

[Figure 18: Create the SOAP over JMS Interface: Service Settings 66](#_Toc482786152)

[Figure 19: Create the SOAP over JMS Interface: New SOAP Binding 66](#_Toc482786153)

[Figure 20: Create the SOAP over JMS Interface: New SOAP Binding 67](#_Toc482786154)

[Figure 21: Create the SOAP over JMS Interface: Transport Type 68](#_Toc482786155)

[Figure 22: Create the SOAP over JMS Interface: JMS Settings 68](#_Toc482786156)

[Figure 23: Create the SOAP over JMS Interface: Queue Name as Property 69](#_Toc482786157)

**Table of Tables**

[Table 1: Acronyms 9](#_Toc446429920)

[Table 2: Issues 10](#_Toc446429921)

1. Document Control
   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. Distribution Information

| **Name** | **Project Role** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* + 1. TIBCO

| **Name** | **Project Role** |
| --- | --- |
| Andrew Dearing | Project Delivery Manager |

* 1. Reference Documents

| **Name** | **Author** |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Document Acceptance

|  |  |
| --- | --- |
| Signed for and on behalf of | |
| Signature of authorised officer: |  |
| Name of authorised officer: |  |
| Witness: |  |
| Date: |  |

1. Introduction

To enable the standardisation of services within TESCO 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.

* 1. Scope

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

* 1. Purpose

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

* 1. Audience

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

* 1. 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 |
| SVN | Apache Subversion source control system |
| XML | Data format: Extended Markup Language |

* 1. Issues

Table 2: Issues

| # | Status | Owner | Description / Resolution |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. Risks

TBC

1. Create a Business Service Provider as an Application

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

* 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.
  + **FlushExceptionEvents**: if ESB System Error Handler events are cached in memory, this process is triggered by a timer repeatedly to flush the cache into the ESB System Error Handler JMS destination.
  + **FlushLogEvents**: if ESB System Audit events are cached in memory, this process is triggered by a timer repeatedly to flush the cache into the ESB System Audit JMS destination.
* 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.
   1. Pre-Requisites

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

* + 1. Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder APP\_BSServiceTemplate.moduleinto the folder **<SVN> / 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>

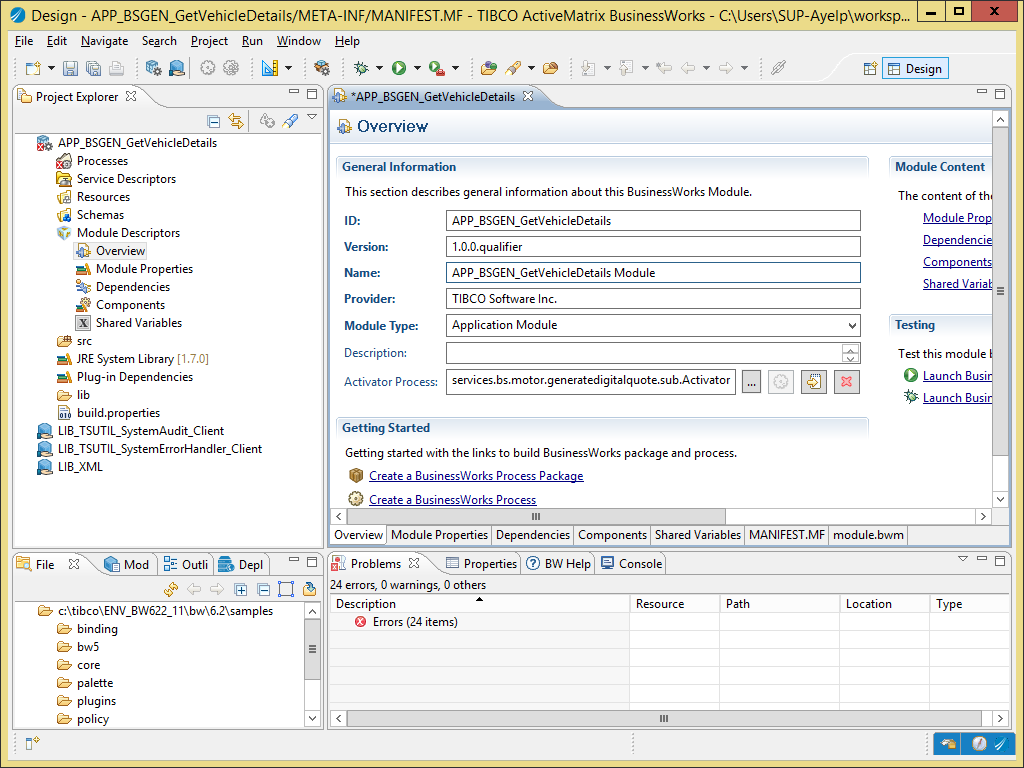
…

With TIBCO BusinesStudio

1. Start TIBCO BusinessStudio
2. In your workspace, import the following modules:
   1. LIB\_XML (from <SVN> / trunk / XML
   2. LIB\_TSUTIL\_SystemAudit\_Client (from <SVN> / trunk / BW / TechnicalServices)
   3. LIB\_TSUTIL\_SystemErrorHandler\_Client (from <SVN> / trunk / BW / TechnicalServices)
   4. APP\_<TYPE><Area>\_<ServiceName> (from <SVN> / trunk / BW / BusinessServices)

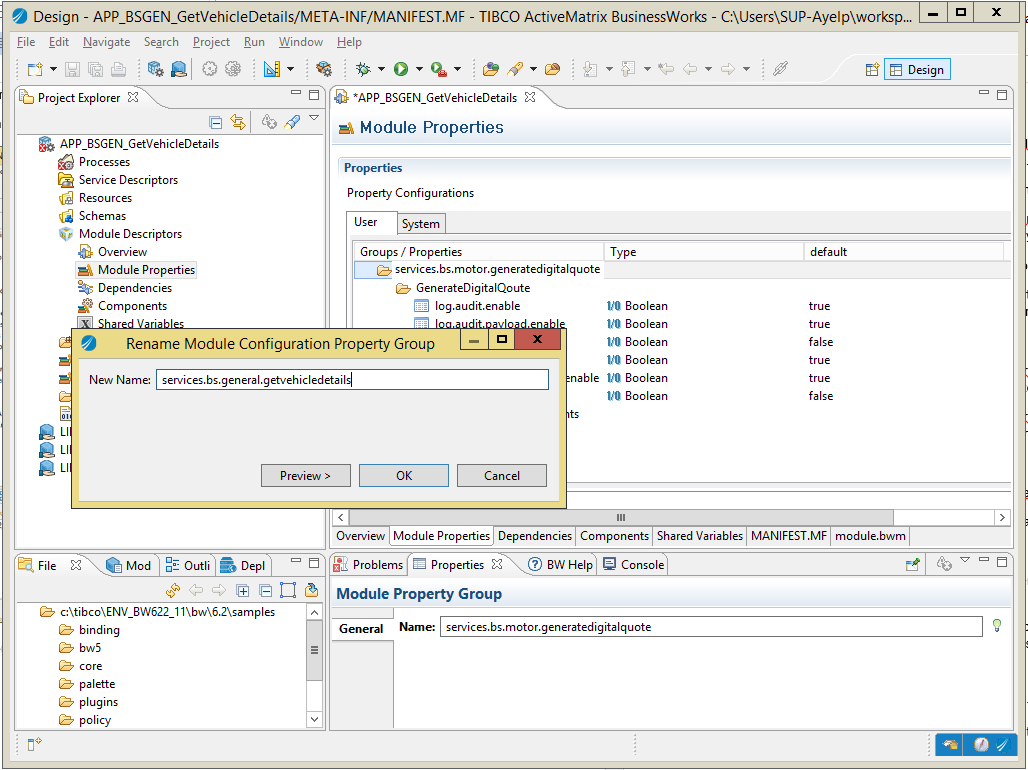
All subsequent changes are done on the APP\_<TYPE><Area>\_<ServiceName> module:

1. Go to the module overview.
2. Change the name into “APP\_BS<Area>\_<ServiceName> Module”:

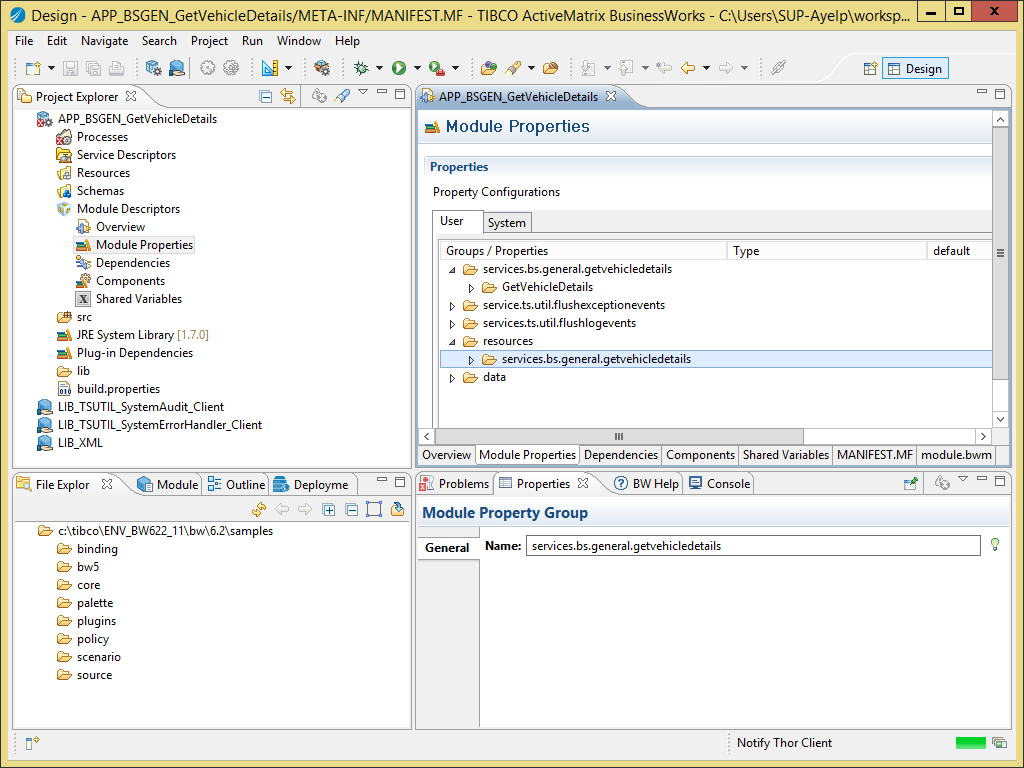


* + 1. Change the Module Properties

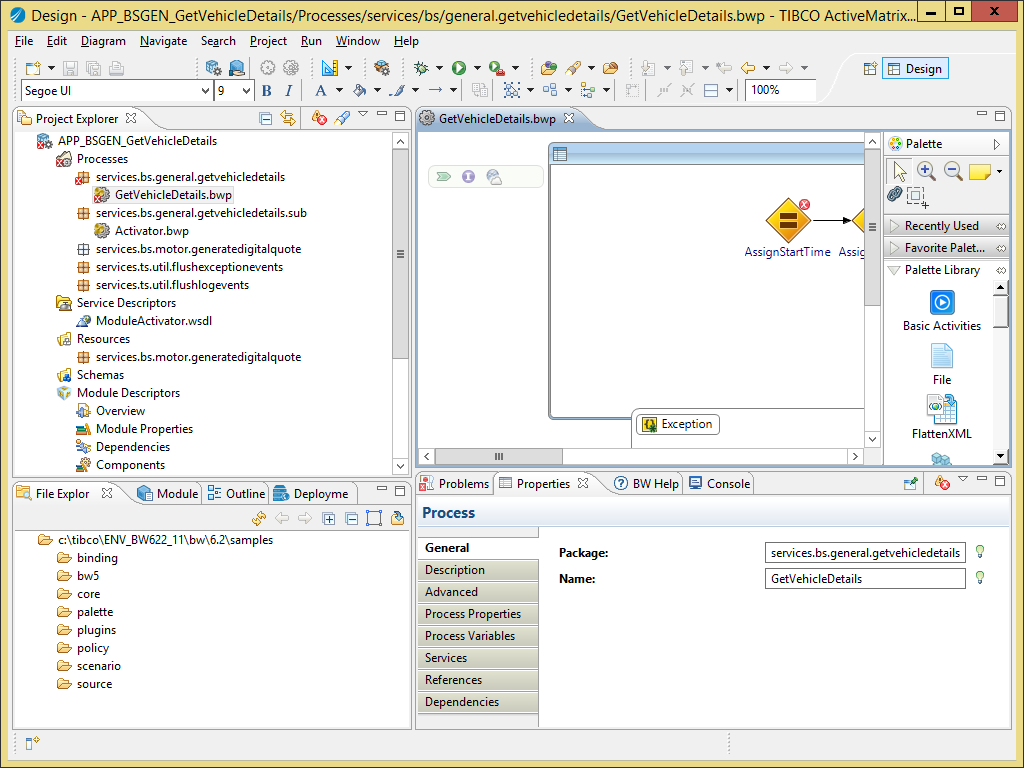
1. Go to the module properties.
2. Select the group “services.bs.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.<type><area>.<servicename>”:



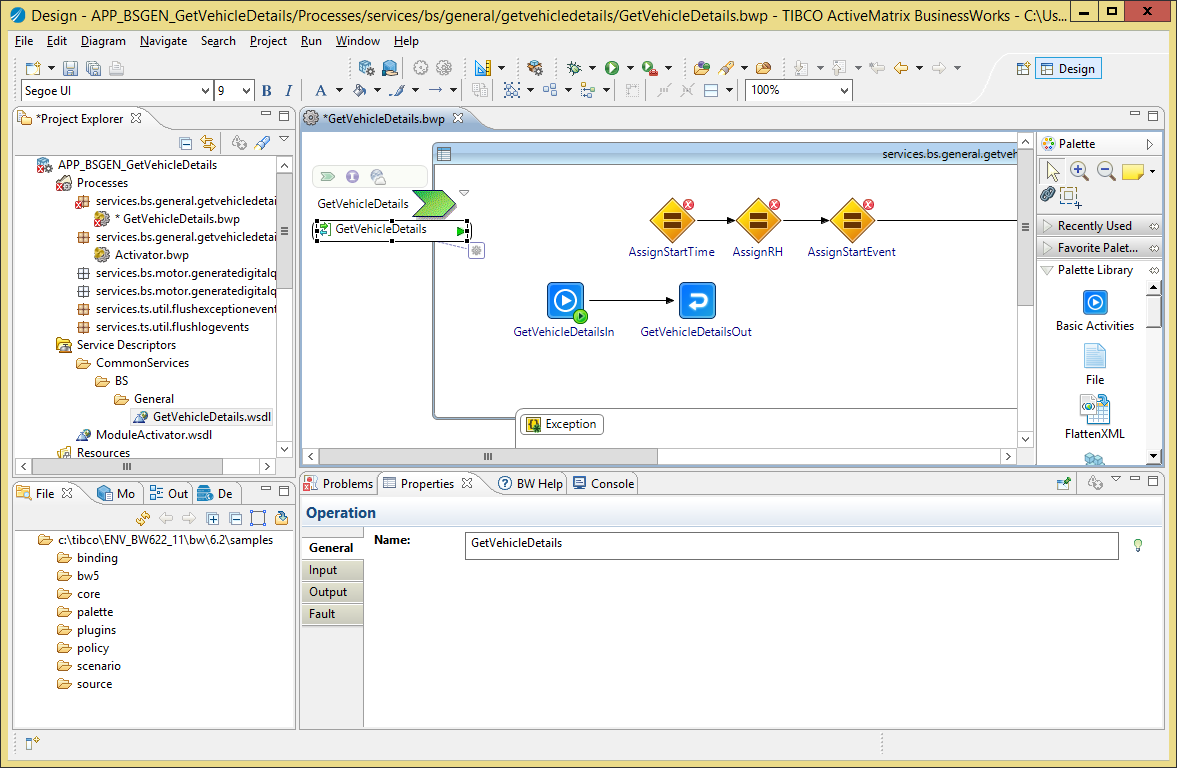
1. In this group, use the same technique to rename the group “ServiceName” into “<ServiceName>”.
2. Select the group “resources / services.bs.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.<type><area>.<servicename>”:



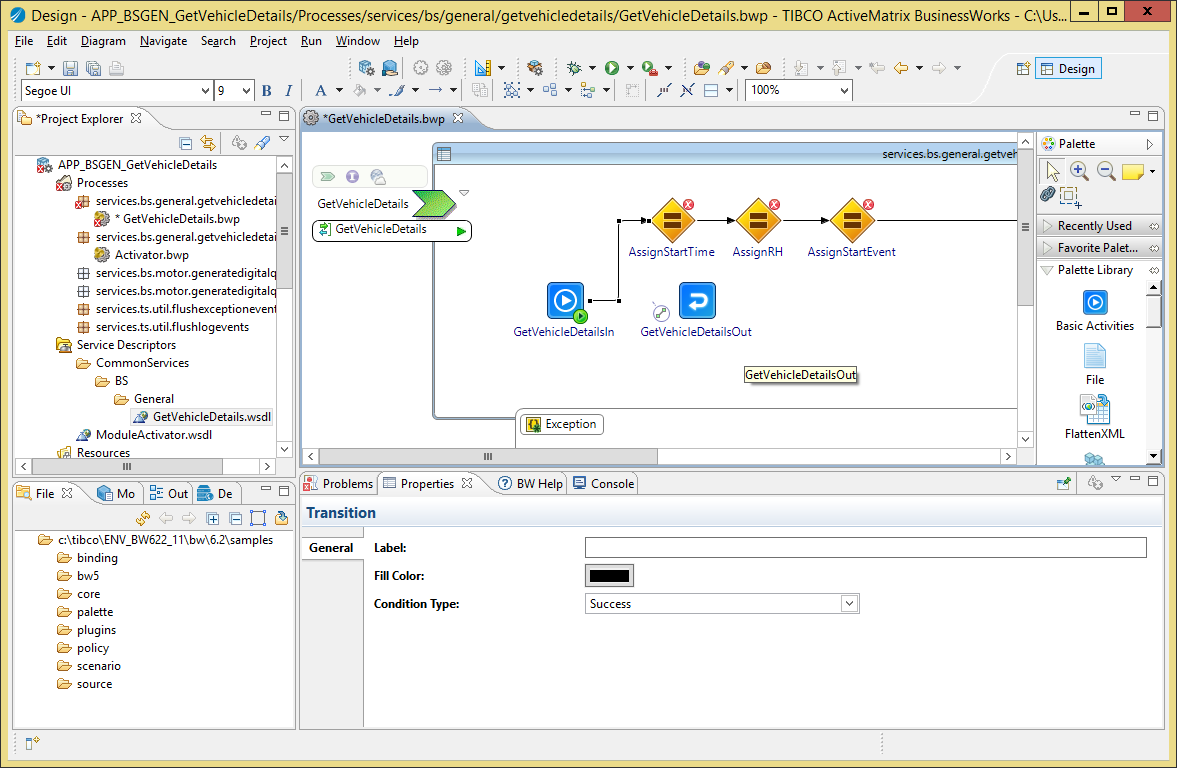
1. In this group, change the value of the HTTP port to the port assigned for the service.
   * 1. Rename the Resources
2. Open the HTTP Server resource. In the properties pane, use the light-bulb icon to change the name into “services.<type><area>.<servicename>.Server-BW01” (only the package name should change).
3. Do the same for the JNDIClient-ESB01 resource.
4. Do the same for the JMSClient-ESB01 resource.
   * 1. Create the Process Packages
5. Create the process package “services.<type>.<area>.<servicename>”.
6. Move the BSService process into it.
7. Create the process package “services.<type>.<area>.<servicename>.sub”.
8. Move the Activator process into it.
9. Delete the remaining …ServiceName… process packages.
   * 1. Change the Service Process
10. Open the process “services.<type>.<area>.<servicename> / BSService”.
11. In the properties, use the light-bulb icon to rename it into “<ServiceName>”:



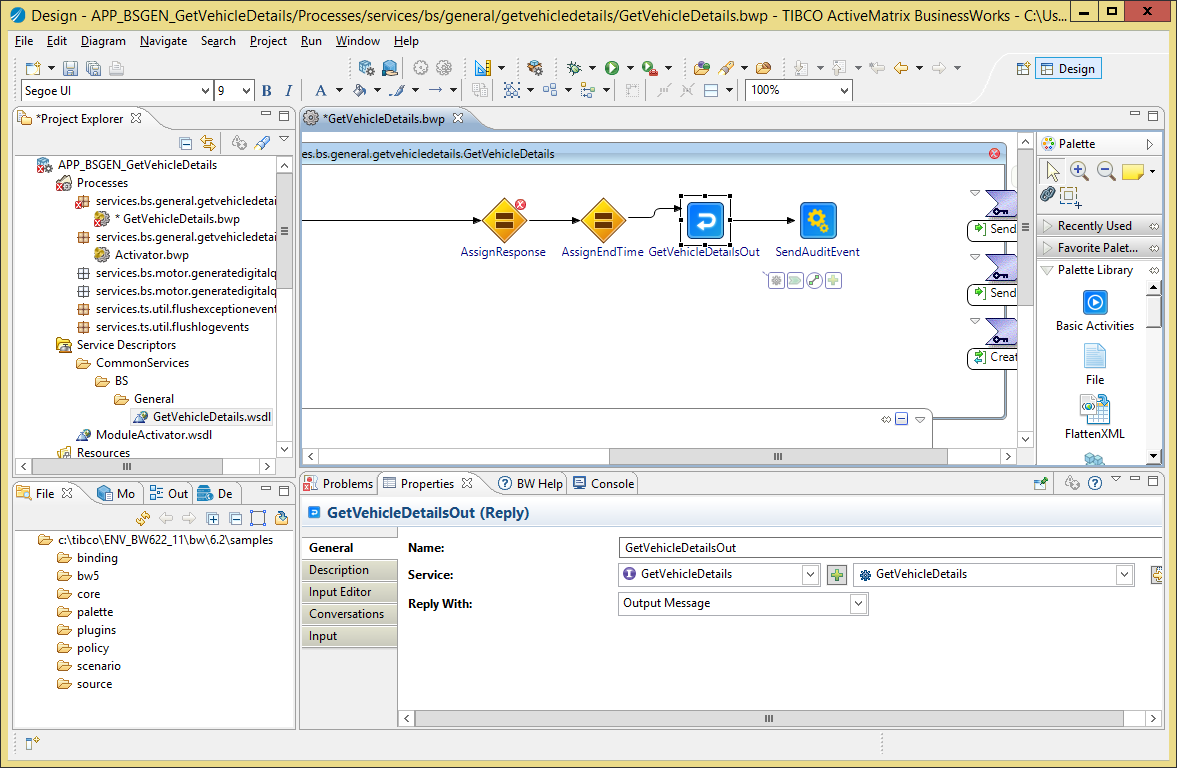
1. Import the XSD and WSDL into the module.
2. Drag and drop the abstract WSDL into the process so the service operation is implemented:



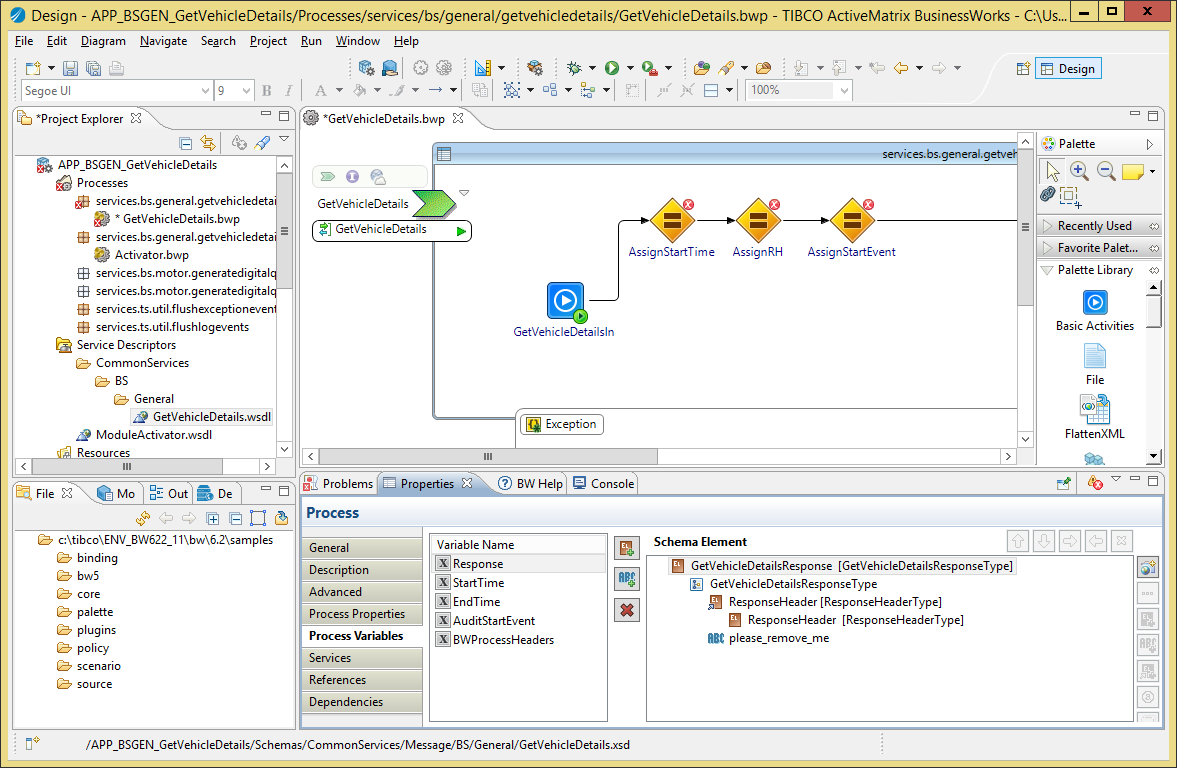
1. Create a transition between the <Operation>In task and “AssignStartTime” task:



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



1. Fix the condition in the transition named “InputError” in the catch-all block as following: $FaultDetails/ActivityName = "<Operation>In".
2. Go the Process Variables. Change the schema of the “Response” variable so it uses the “<Operation>Response” element from the service XSD:



1. Save your changes.
   * 1. Fix the Process Mappings
        1. AssignRH

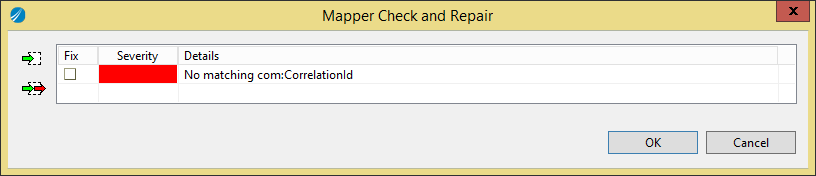
This task prepares the message header that can be passed into calls to other ESB services or returned in the reply or fault messages.

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

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

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

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

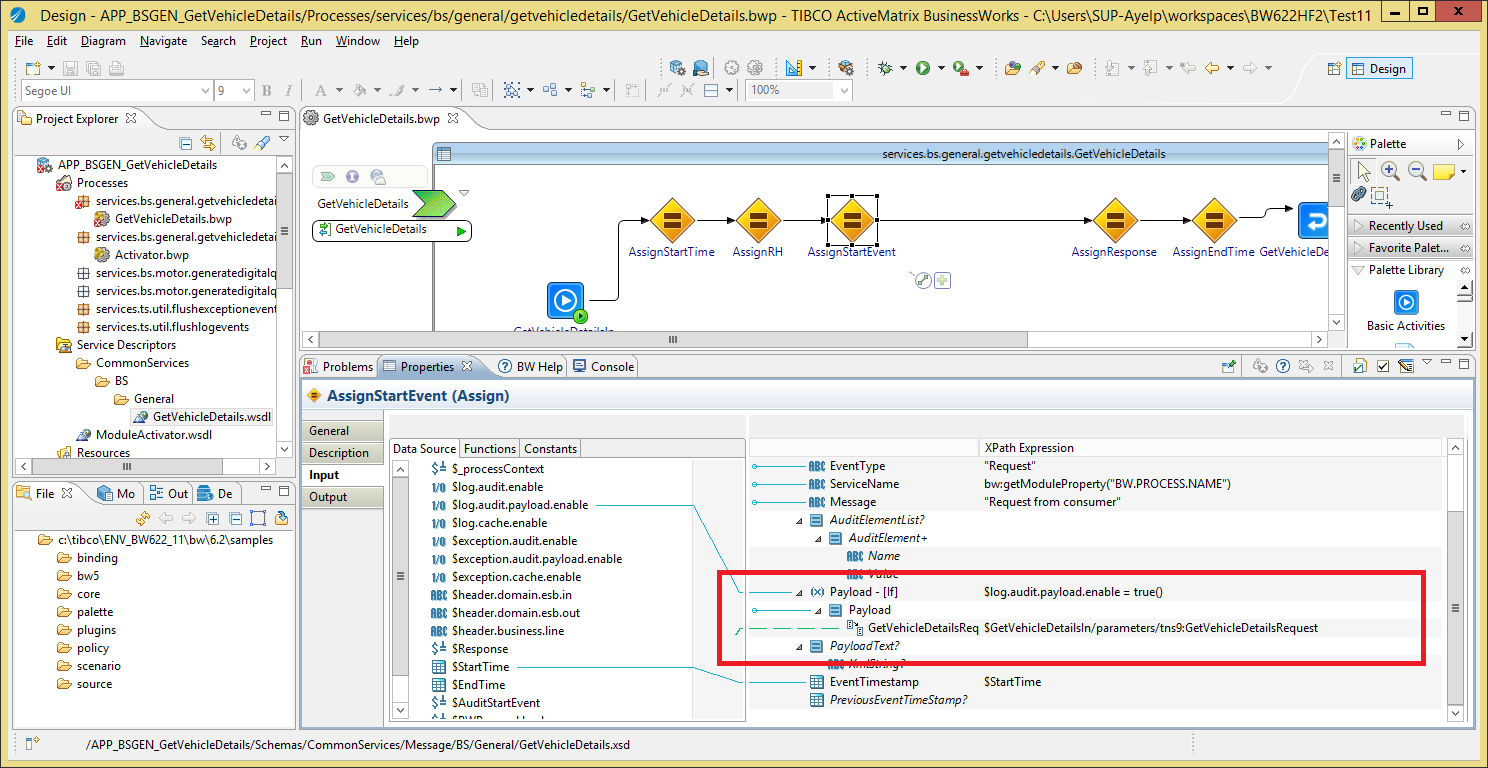


* + - 1. AssignStartEvent

This task prepare 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()”:

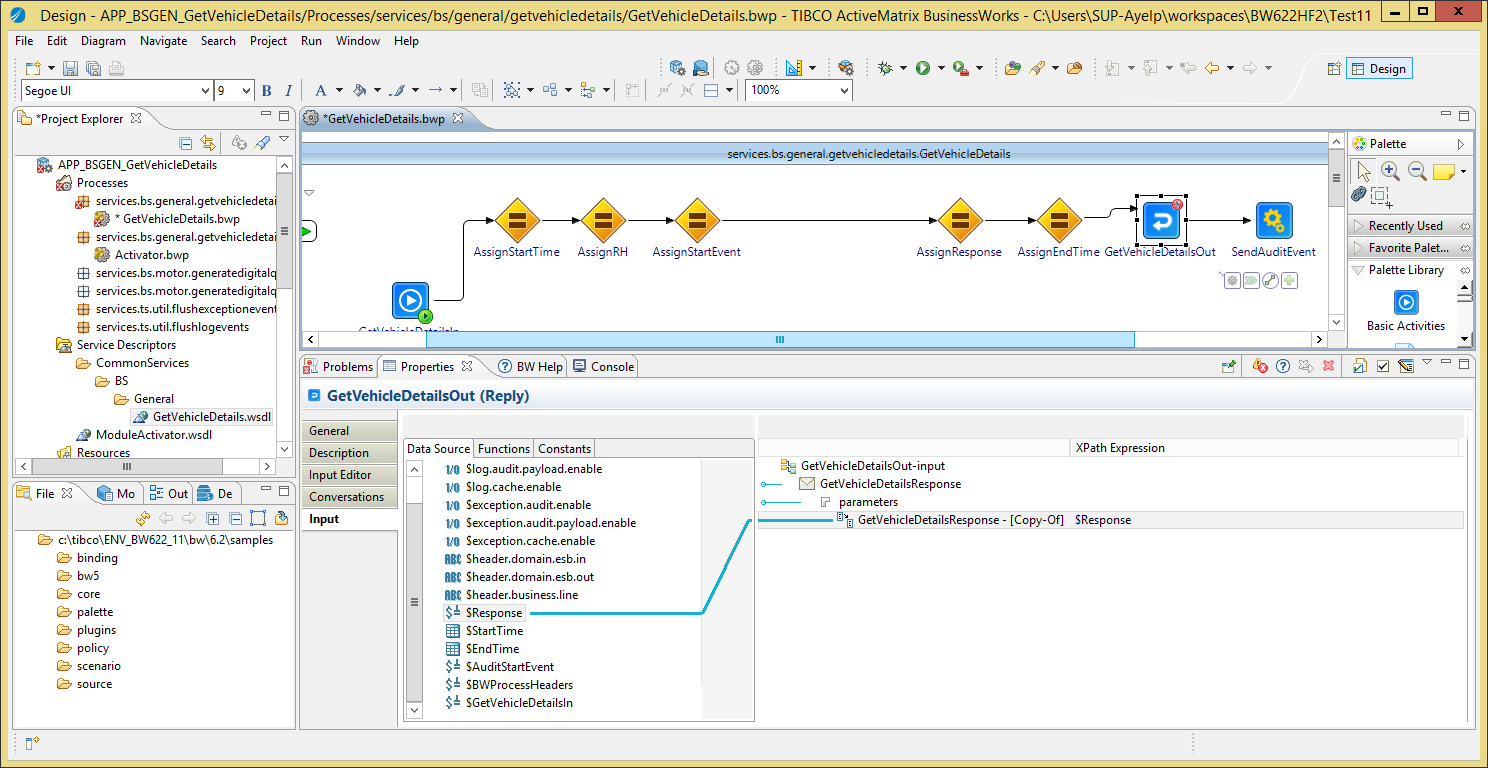


* + - 1. <Operation>Out

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

Fix the mapping as following:

1. <Operation>Response/parameters/<Operation>Response should be a copy-of the $Response XML element:

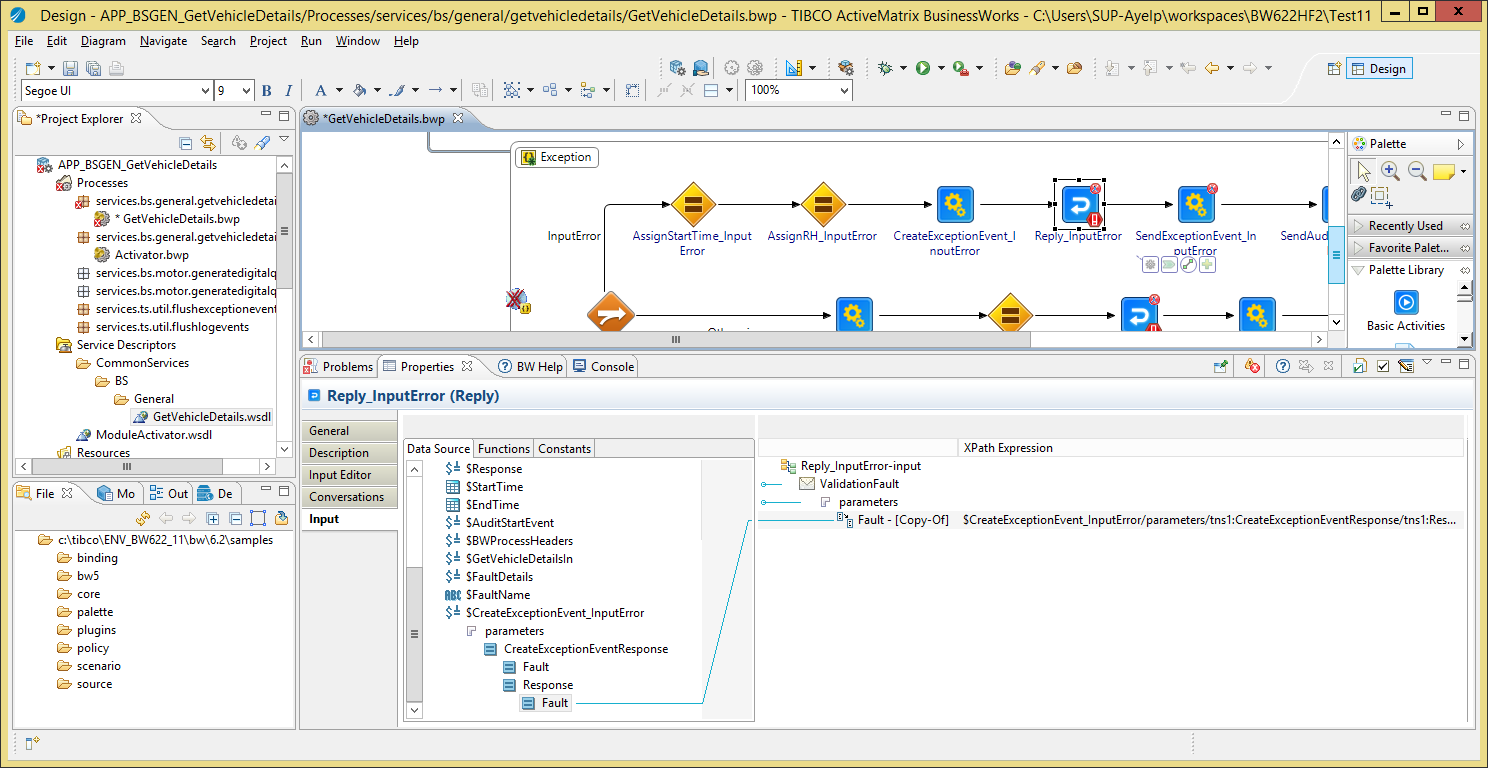


* + - 1. Reply\_InputError

This tasks returns a ValidationFault SOAP fault to the consumer.

Fix the configuration of the Reply\_InputError task (select the service, operation and “ValidationFault”).

1. Fix the mapping of the Reply\_InputError task so ValidationFault element is a copy of the CreateExceptionEvent\_InputError / Response / ValidationFault element:



* + - 1. Reply\_Fault

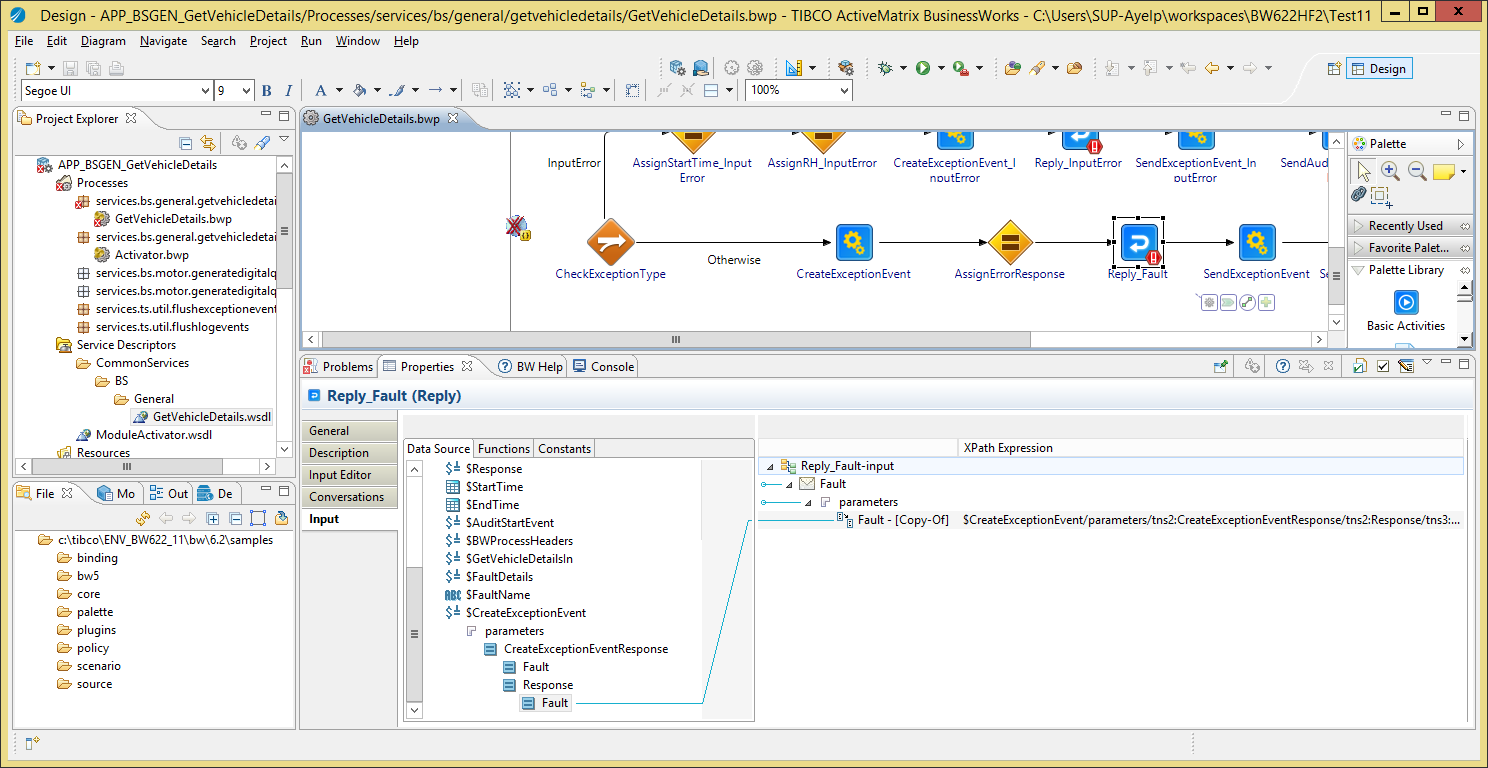
This task returns the fault prepared by CreateExceptionEvent as Fault for the service, in case it contains an XML Fault element.

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:



* + - 1. TimeoutFault

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

Fix the configuration of the TimeoutFault task:

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

Fix the mapping of the TimeoutFault task so:

1. TimeoutFault element is a copy of the CreateExceptionEvent / Response / TimeoutFault element.
   * + 1. BackEndFault

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

Fix the configuration of the BackEndFault task:

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

Fix the mapping of the BackEndFault task so:

1. BackEndFault element is a copy of the CreateExceptionEvent / Response / BackEndFault element.
   * + 1. 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.
   * + 1. AuthenticationFault

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

Fix the configuration of the AuthenticationFault task:

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

Fix the mapping of the AuthenticationFault task so:

1. AuthenticationFault element is a copy of the CreateExceptionEvent / Response / AuthenticationFault element.
   * 1. Create the SOAP Binding
2. TBC
   1. Implement the Service
      1. Audit Fields

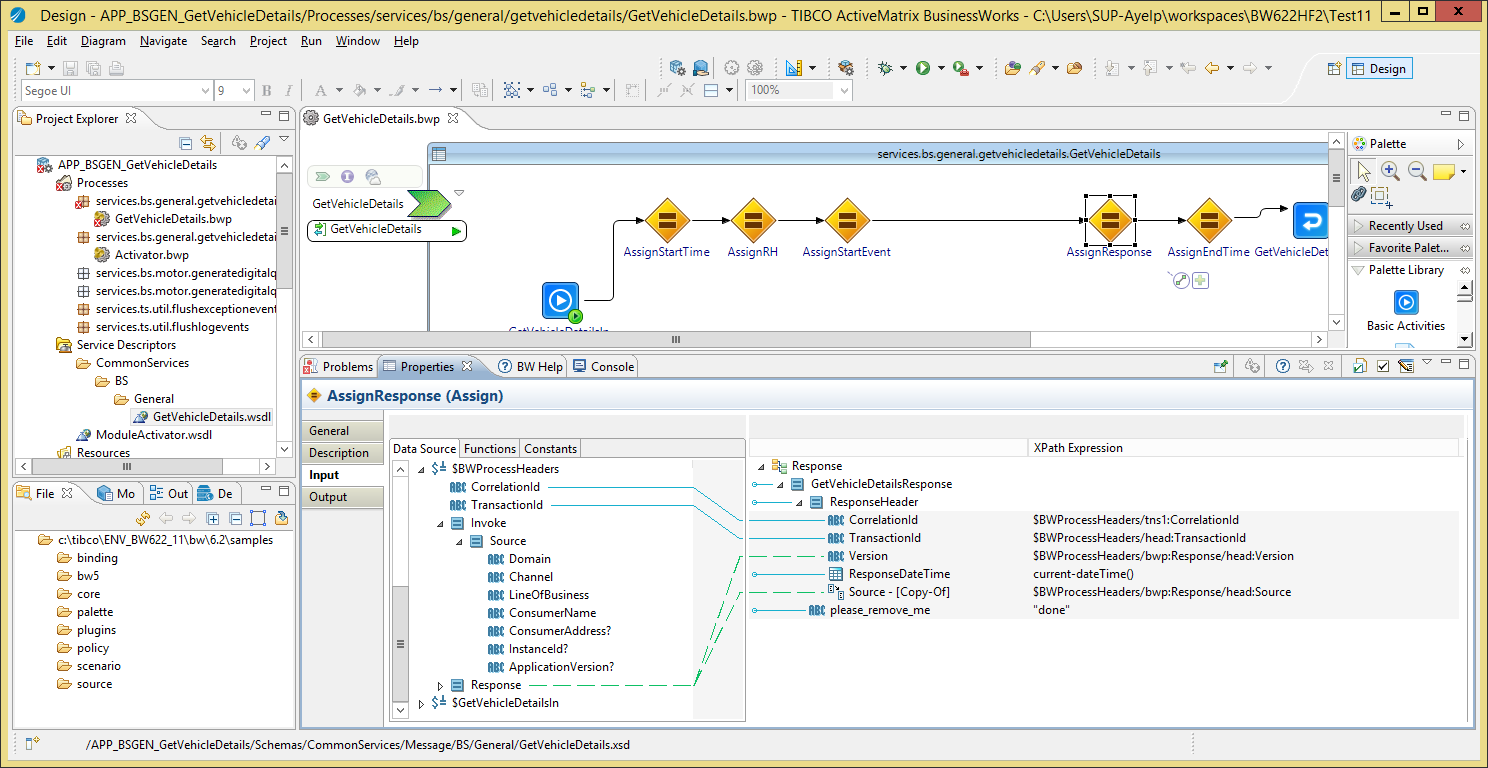
TBC

* + 1. Business Logic

1. Implement the Service: put the logic task between the “AssignStartEvent” and “AssignResponse” tasks.
   * 1. AssignResponse
2. 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:

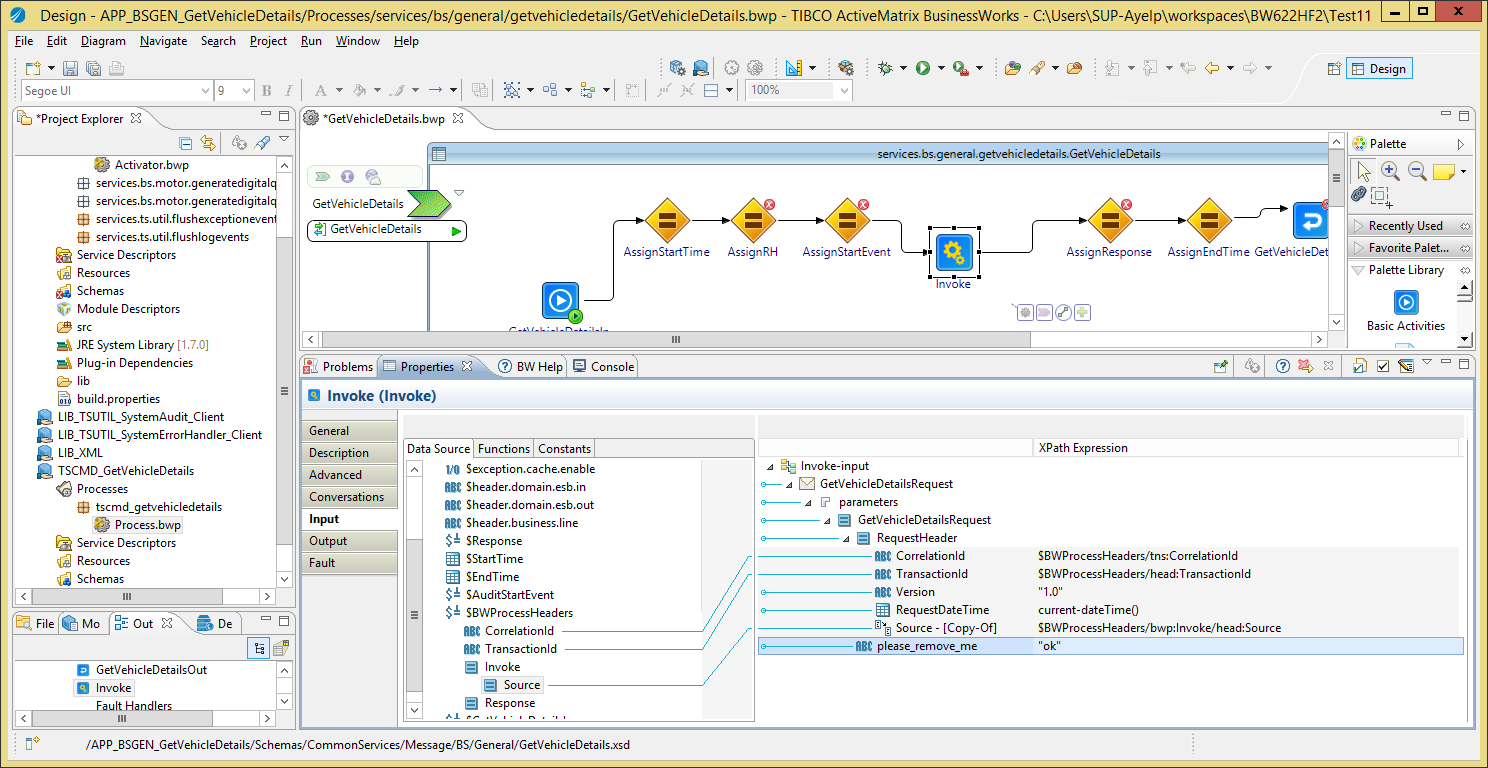
* 1. CorrelationId: $BWProcessHeaders / CorrelationId
  2. TransactionId: $BWProcessHeaders / TransactionId
  3. Version: $BWProcessHeaders / Response / Version
  4. ResponseDateTime: current-dateTime()
  5. Source: copy of $BWProcessHeaders / Response / Source



* + 1. Error Handling

1. Depending on the design, some services may return a valid response in case of fault. In such case:
   1. edit the mapping AssignErrorResponse task
   2. Change the configuration of the Reply\_Fault task so it return a response and the mapping use the $Response process variable.
   3. You can do the same for other branches of fault if certain faults must return a valid reply and not a fault:
      1. Add an Assign task to assign values into the $Response variable.
      2. Change the configuration and mapping of the XXXFault task so it return the $Response as valid response instead of a fault.
      3. Invoke another ESB service
2. If you invoke another ESB service (BS or TS), you MUST map the RequestHeader element as following:
   1. CorrelationId: $BWProcessHeaders / CorrelationId
   2. TransactionId: $BWProcessHeaders / TransactionId
   3. Version: The version of the invoked service, for example “1.0”.
   4. RequestDateTime: current-dateTime()
   5. Source: copy of $BWProcessHeaders / Invoke / Source

In this screenshot we invoke TSGEN\_GetVehicleDetails TS service:



* + 1. 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”:

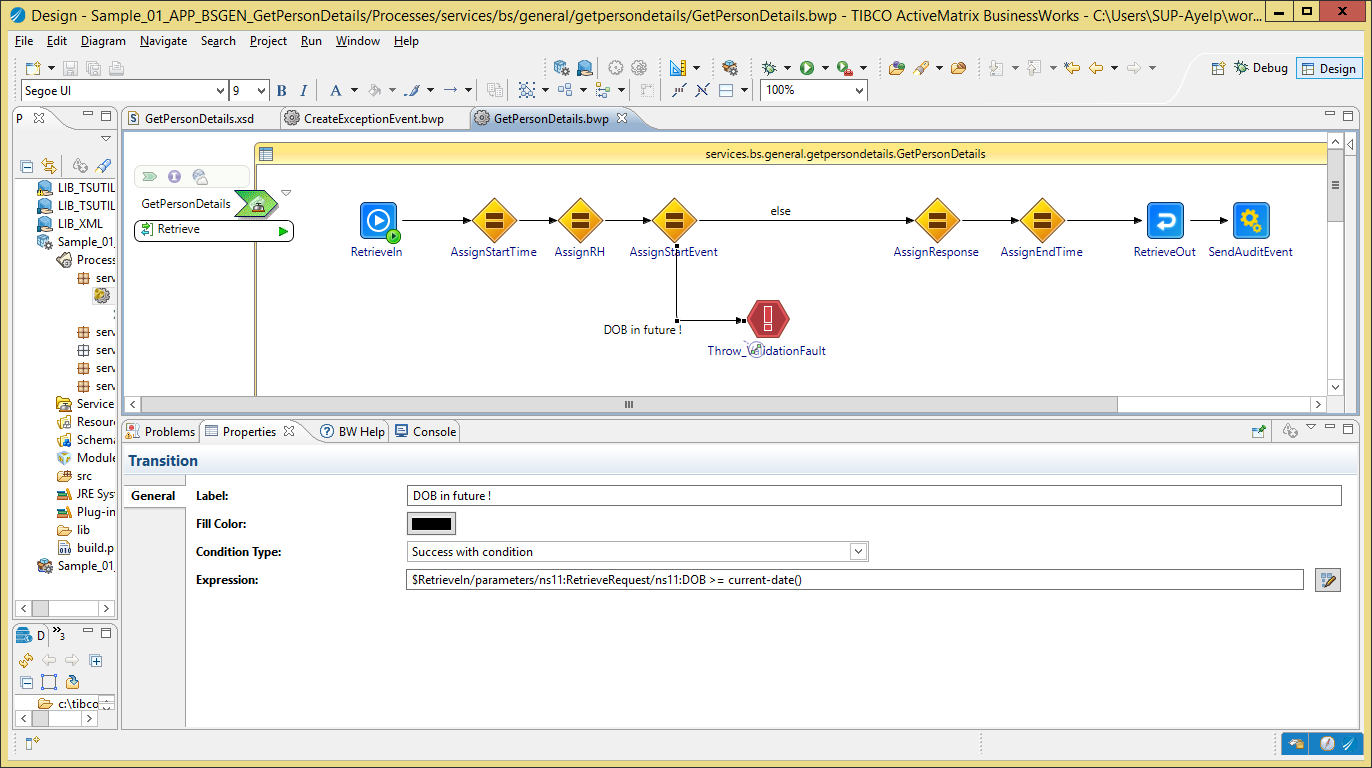


Figure 1: Throw\_ValidationFault sample

1. Change the task “Throw\_ValidationFault” as following:
2. In “Input Editor”, add the XML element “ThrowableValidationFault” from the “Schemas / CommonServices / Data / Message / Internal / Throwable” XSD.
3. Change the input mapping as following:
4. ExceptionTimestamp: current-dateTime()
5. ExceptionDescription: a description of the issue, for example concat(“Car Reg is incorrect: “, $xxx). When you mention an input element value is incorrect, always put its value in the error description (e.g. $xxx).

See the following example (to see it, unzip it into your trunk/BW/BusinessServices folder and import it into your Eclipse workspace, where you already have LIB\_XML, LIB\_SystemAudit\_Client.2 and LIB\_SystemErrorHandler\_Client.2):



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.
* See the following example (to see it, unzip it into your trunk/BW/BusinessServices folder and import it into your Eclipse workspace, where you already have LIB\_XML, LIB\_SystemAudit\_Client.2 and LIB\_SystemErrorHandler\_Client.2):



* 1. 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.

1. 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.

* 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:***

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.
   1. Pre-Requisites

The XSD and WSDL have been created with the SchemaCreator tool explained in the “The AA - ESB - How To - Create Service Schemas” document.

* 1. Procedure
     1. Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “LIB\_BSServiceTemplate.module” into the folder **<SVN> / 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

1. Start TIBCO BusinessStudio
2. In your workspace, import the following modules:
   1. LIB\_XML (from <SVN> / trunk / XML
   2. LIB\_TSUTIL\_SystemAudit\_Client (from <SVN> / trunk / BW / TechnicalServices)
   3. LIB\_TSUTIL\_SystemErrorHandler\_Client (from <SVN> / trunk / BW / TechnicalServices)
   4. ~LIB\_BS<Area>\_<ServiceName> (from <SVN> / trunk / BW / BusinessServices)

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

1. Go to the module overview.
2. Change the name into “LIB\_BS<Area>\_<ServiceName> Module”:
   * 1. 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.

* 1. 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.

1. 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.

* 1. Template Overview

The template module “APP\_ServiceTemplate.module” (renamed as APP\_BSServiceTemplate.module in version 0.4.3) 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:***

1. 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.
   1. Pre-Requisites

The XSD and WSDL have been created with the SchemaCreator tool explained in the “The AA - ESB - How To - Create Service Schemas” document.

* 1. Procedure
     1. Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “LIB\_TSServiceTemplate.module” into the folder **<SVN> / 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:
   1. LIB\_XML (from <SVN> / trunk / XML
   2. LIB\_TSUTIL\_SystemAudit\_Client (from <SVN> / trunk / BW / TechnicalServices)
   3. LIB\_TSUTIL\_SystemErrorHandler\_Client (from <SVN> / trunk / BW / TechnicalServices)
   4. LIB\_TS<Area>\_<ServiceName> (from <SVN> / trunk / BW / TechnicalServices)

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

1. Go to the module overview.
2. Change the name into “LIB\_<TYPE><Area>\_<ServiceName> Module”:

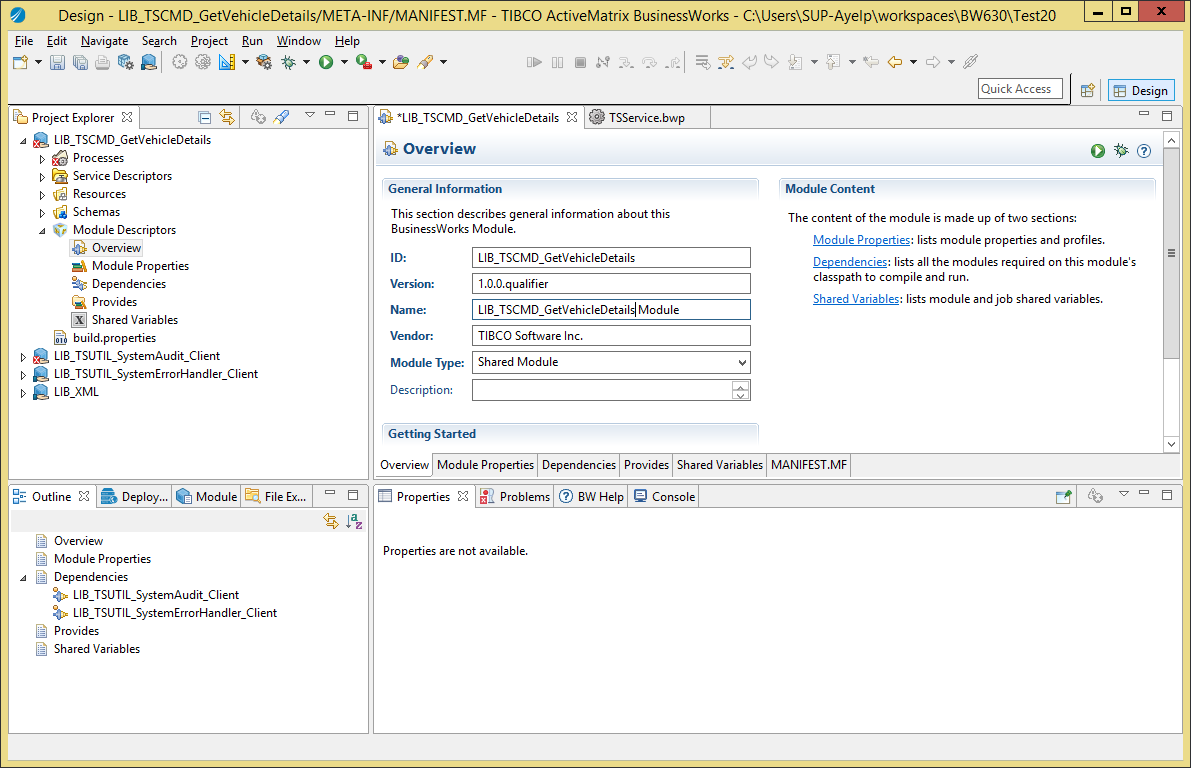


Figure 2: TS Service Provider Shared Module: Module Name

* + 1. 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>”:

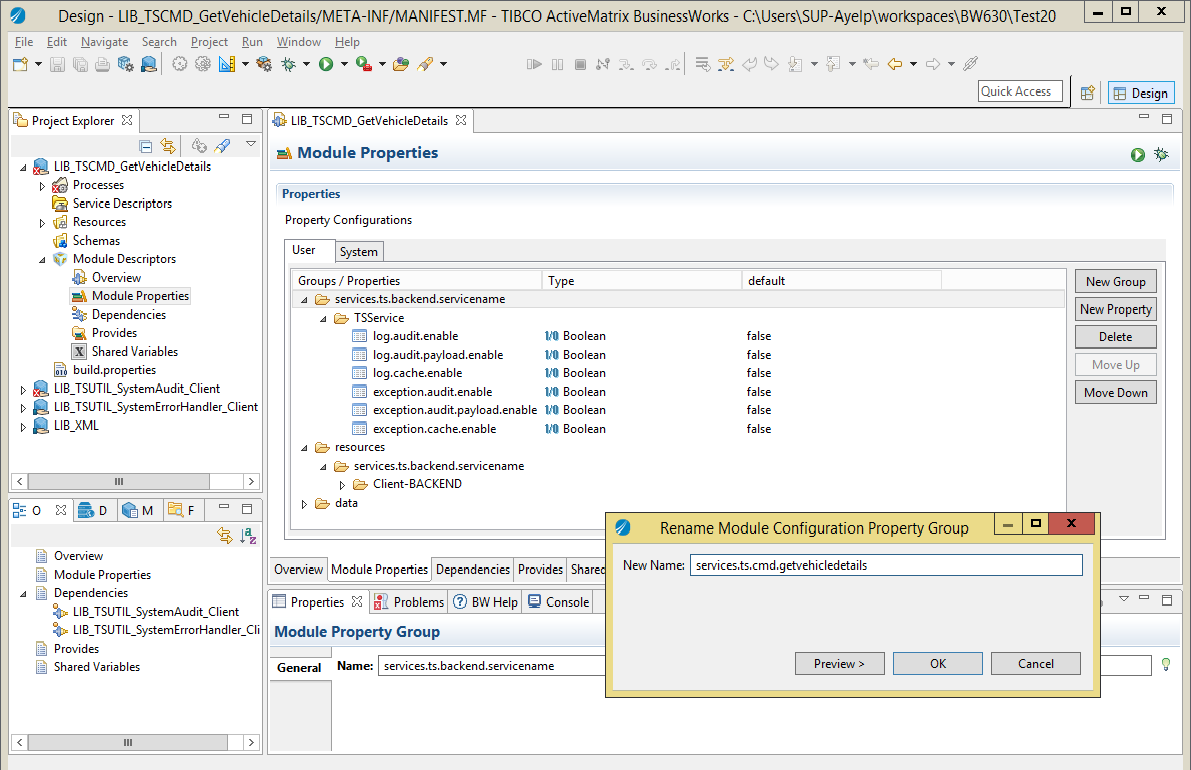


Figure 3: TS Service Provider Shared Module: Module Properties

1. In this group, use the same technique to rename the group “TSService” into “<ServiceName>” (for example “GetVehicleDetails”).
2. 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>”:

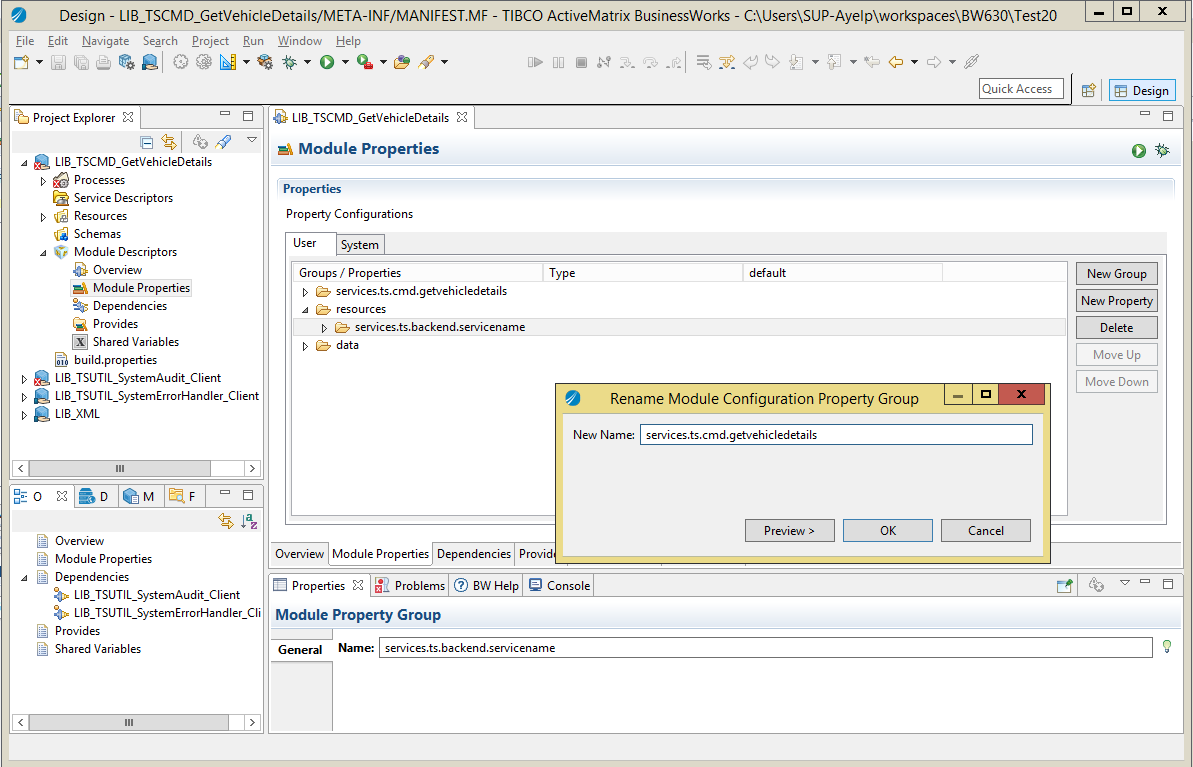


Figure 4: TS Service Provider Shared Module: Resource Properties

* + 1. 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.
   * 1. Change the Service Process
6. Open the process “services.ts.<area>.<servicename> / TSService”.
7. In the properties, use the light-bulb icon to rename it into “<ServiceName>”:

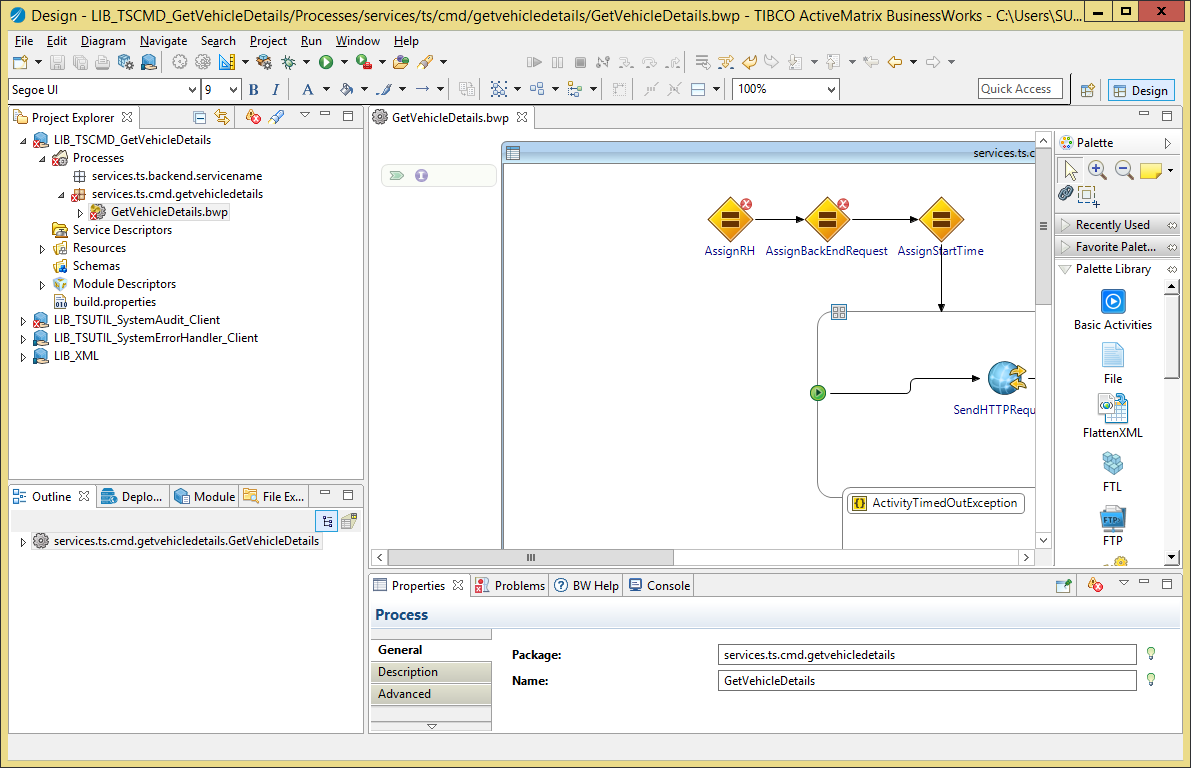


Figure 5: TS Service Provider Shared Module: Process Name

1. Import the XSD and WSDL of the TS Service into the module as mentioned in “The AA - ESB - How To - Create Service Schemas” document.
2. Drag and drop the abstract WSDL into the process as mentioned in “The AA - ESB - How To - Create Service Schemas” document so the service operation is implemented:

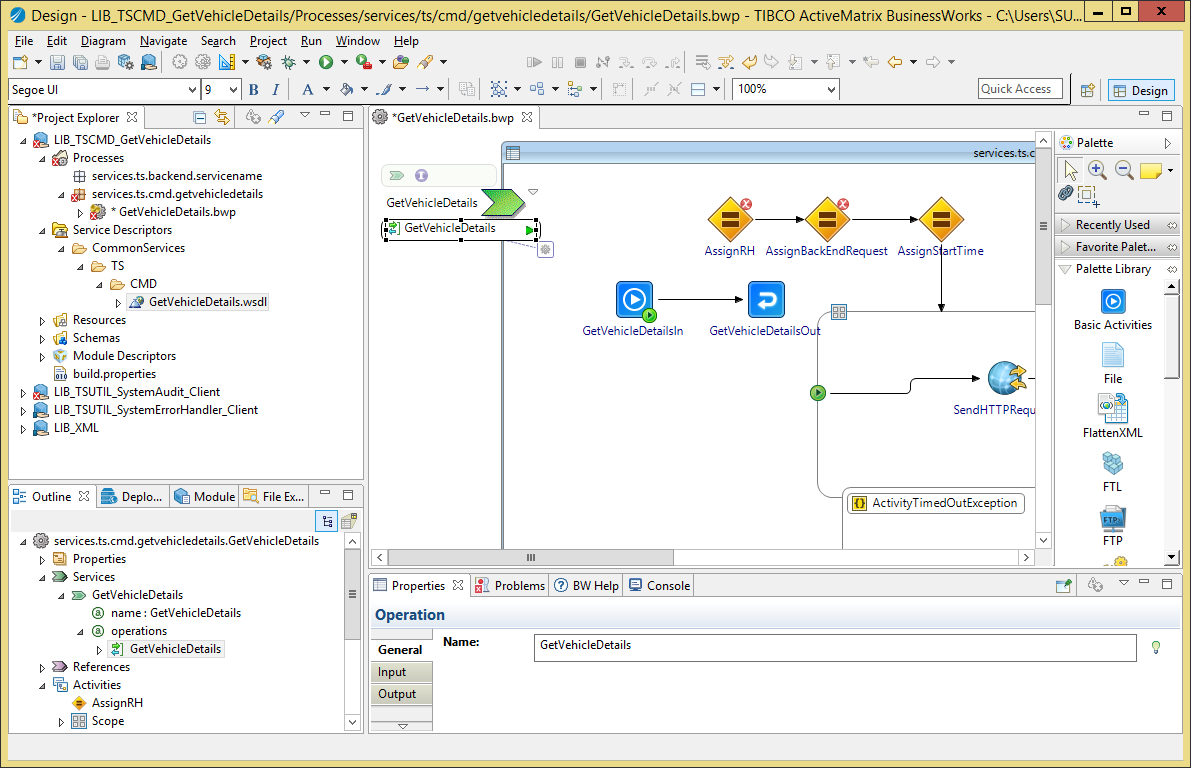


Figure 6: TS Service Provider Shared Module: Implement Service Operation

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

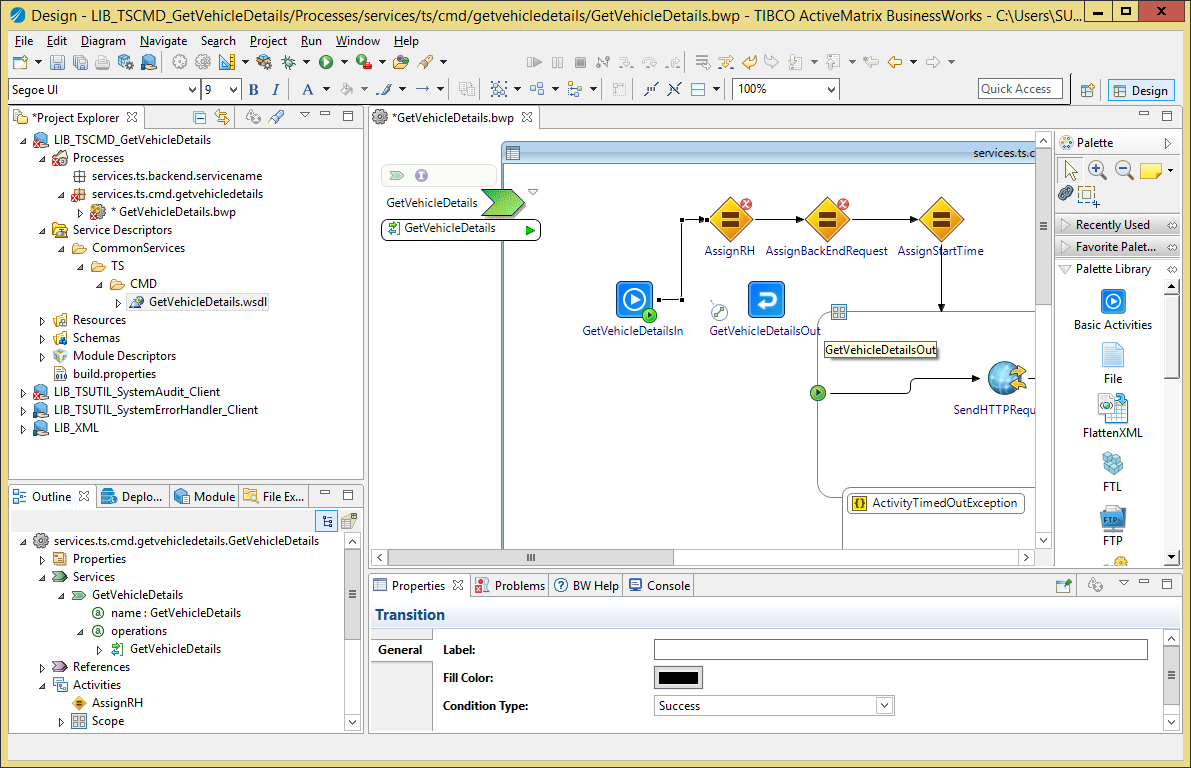


Figure 7: TS Service Provider Shared Module: Change input transition

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

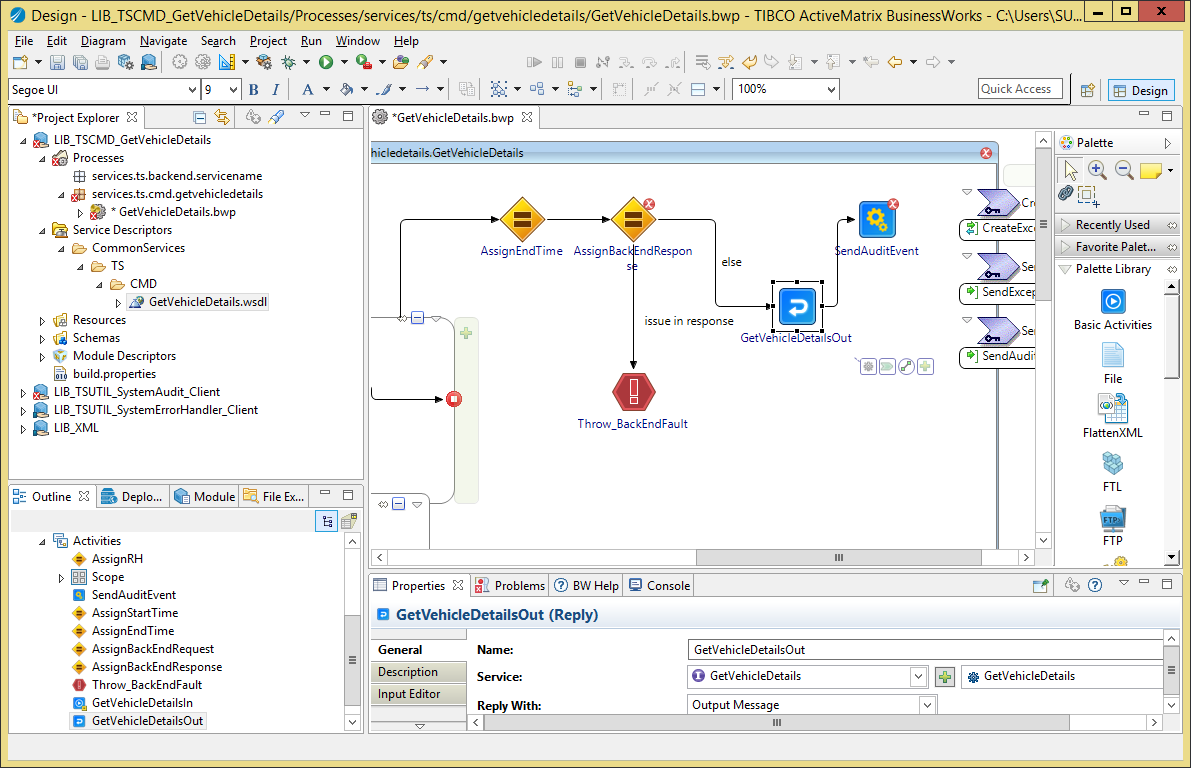


Figure 8: TS Service Provider Shared Module: Reply Transition

1. Save your changes.
   * 1. Fix the Process Mappings

**DO NOT CHANGE MAPPINGS NOT MENTIONED IN THIS SECTION**

* + - 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

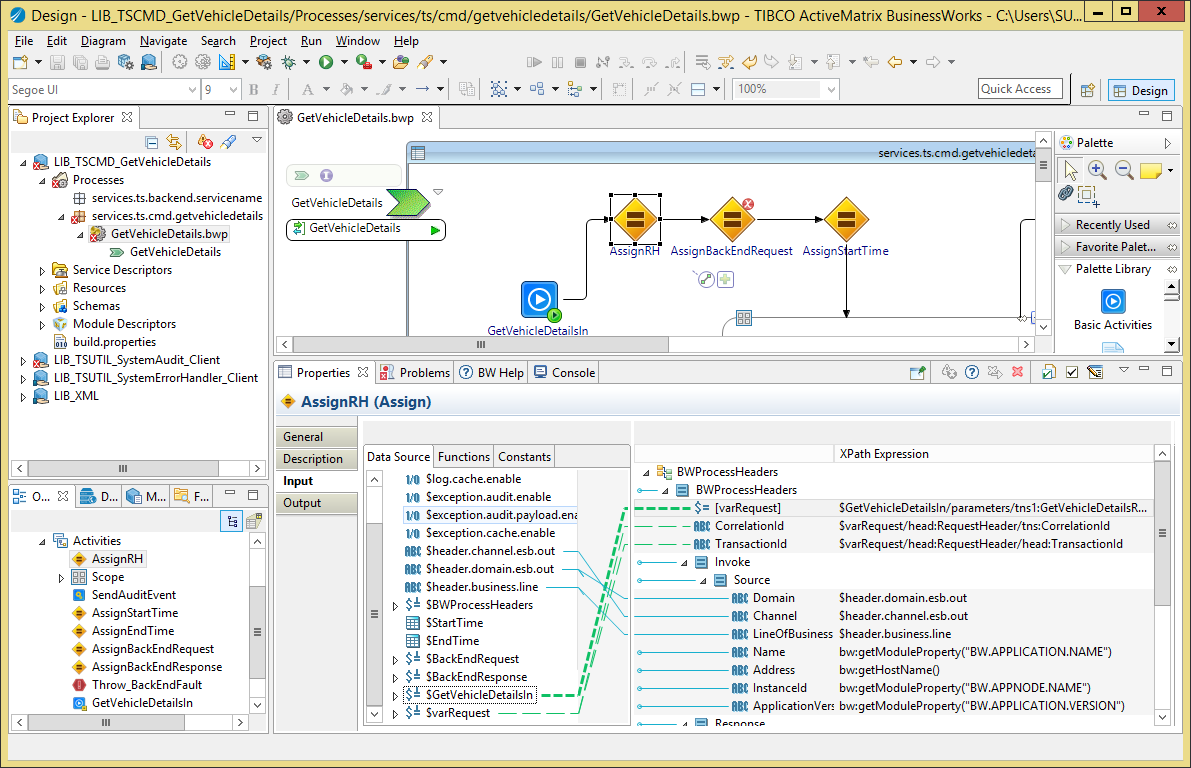


Figure 9: TS Service Provider Shared Module: AssignRH mapping

* + - 1. 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:

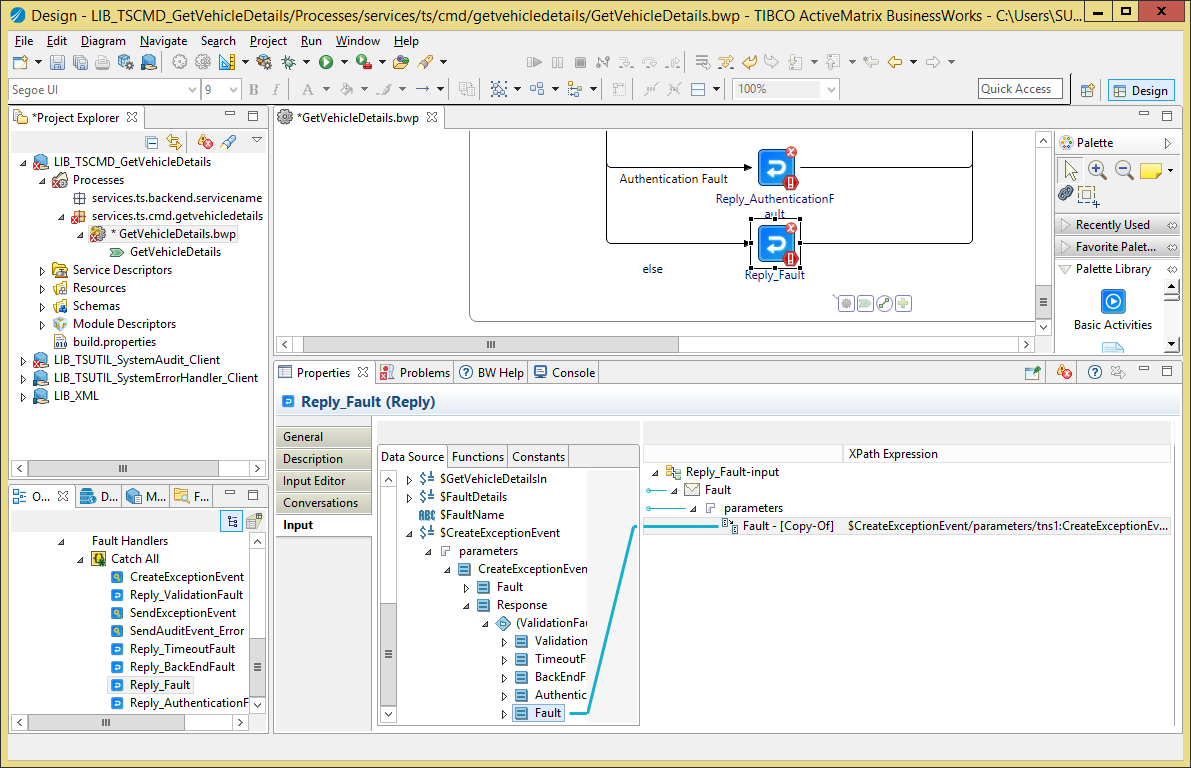


Figure 10: TS Service Provider Shared Module: Reply\_Fault mapping

* + - 1. Reply\_TimeoutFault

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

Fix the configuration of the TimeoutFault task:

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

Fix the mapping of the TimeoutFault task so:

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

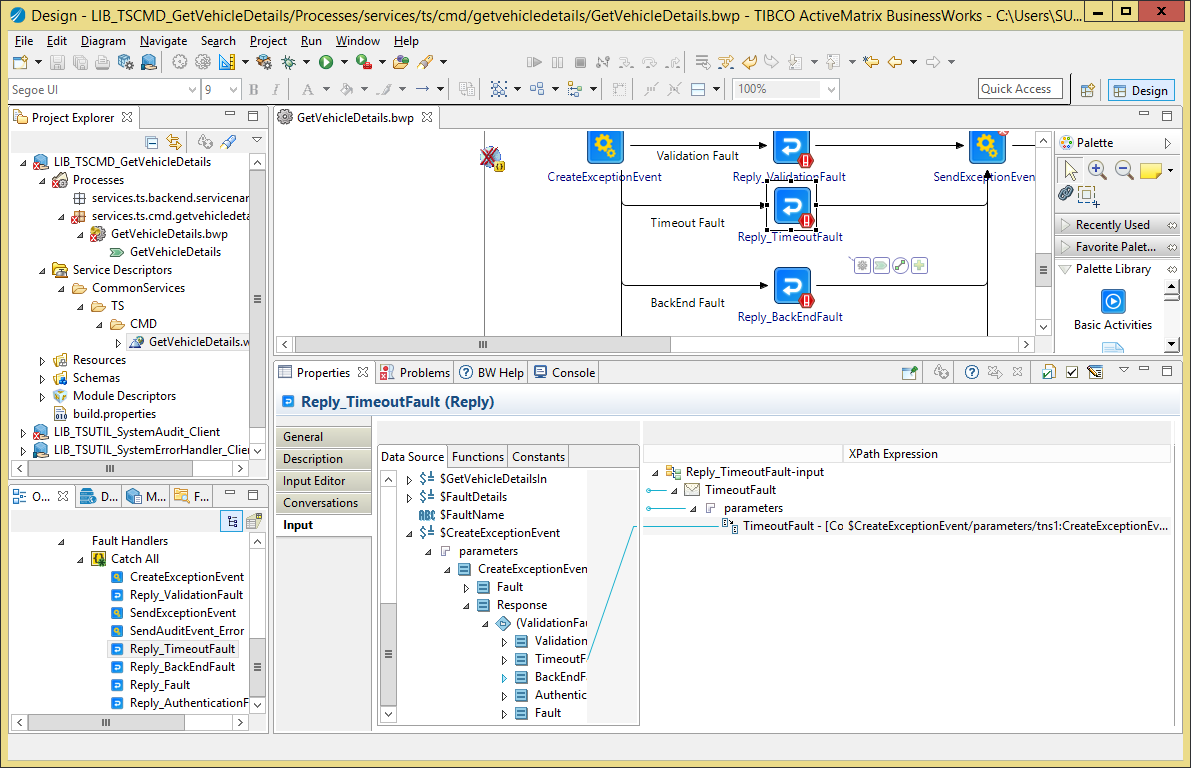


Figure 11: TS Service Provider Shared Module: Reply\_TimeoutFault mapping

* + - 1. Reply\_BackEndFault

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

Fix the configuration of the BackEndFault task:

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

Fix the mapping of the BackEndFault task so:

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

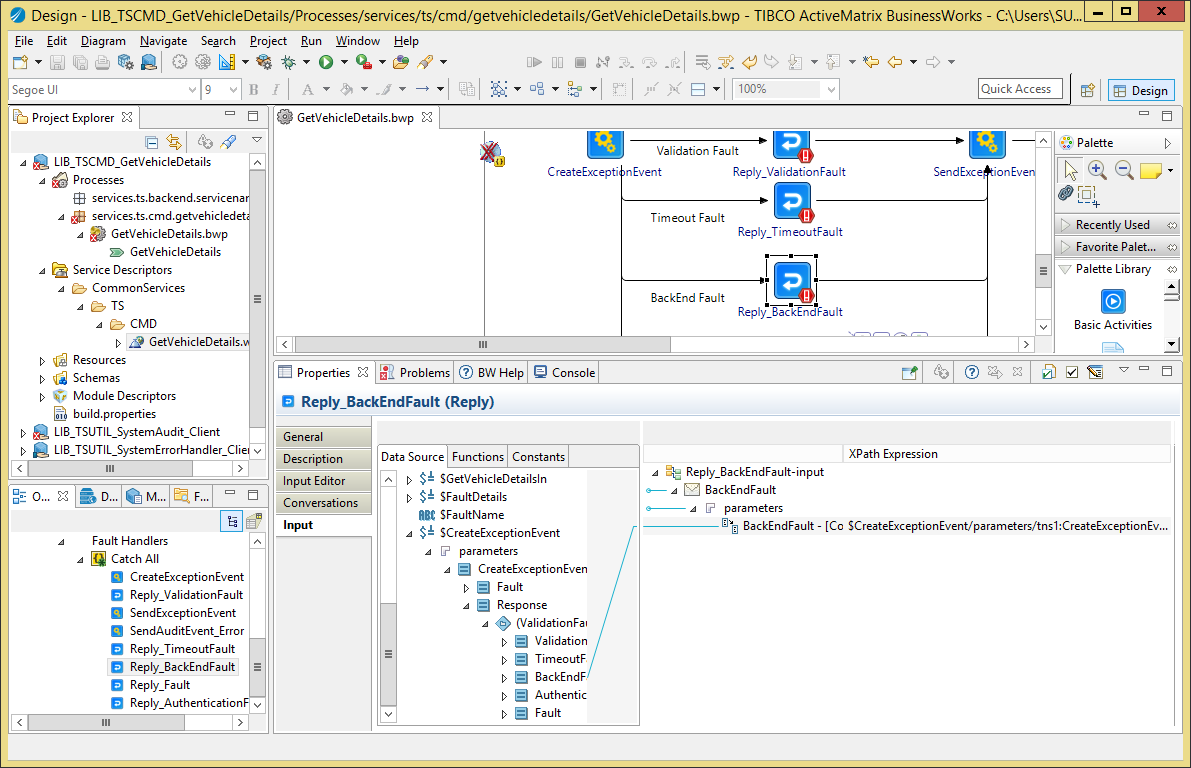


Figure 12: TS Service Provider Shared Module: Reply\_BackEndFault mapping

* + - 1. 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:

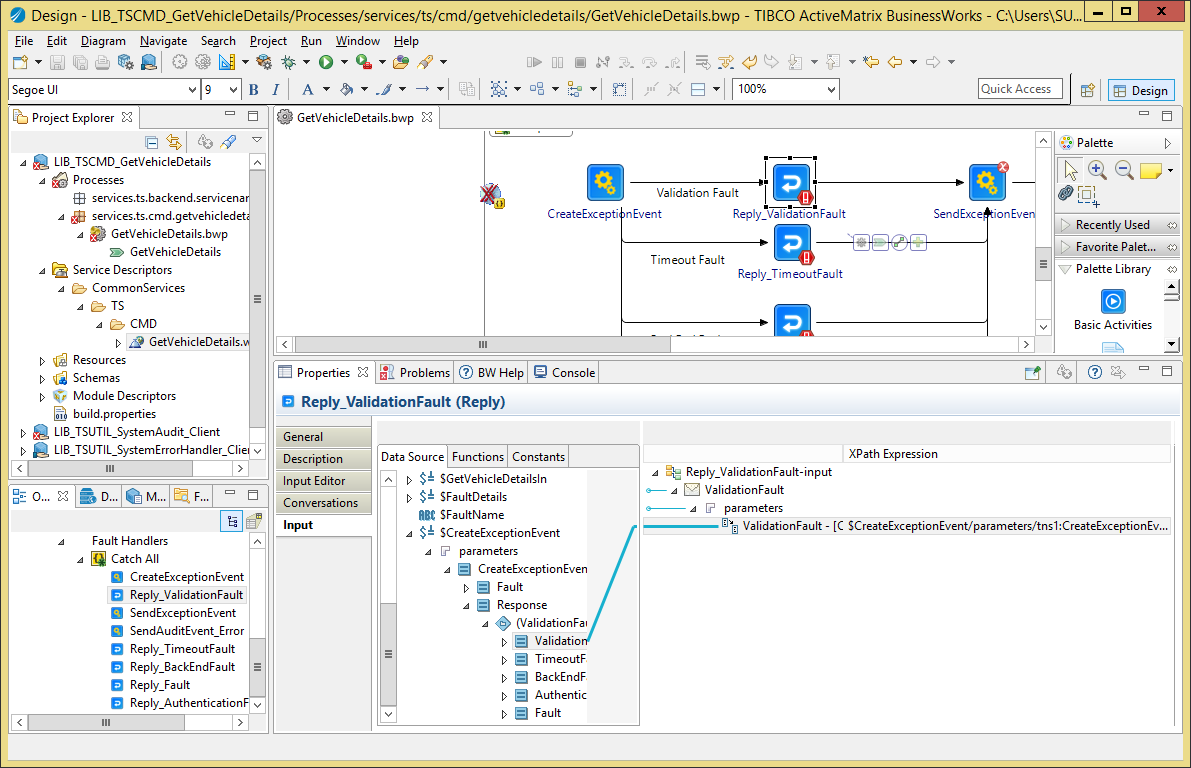


Figure 13: TS Service Provider Shared Module: Reply\_ValidationFault mapping

* + - 1. Reply\_AuthenticationFault

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

Fix the configuration of the AuthenticationFault task:

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

Fix the mapping of the AuthenticationFault task so:

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

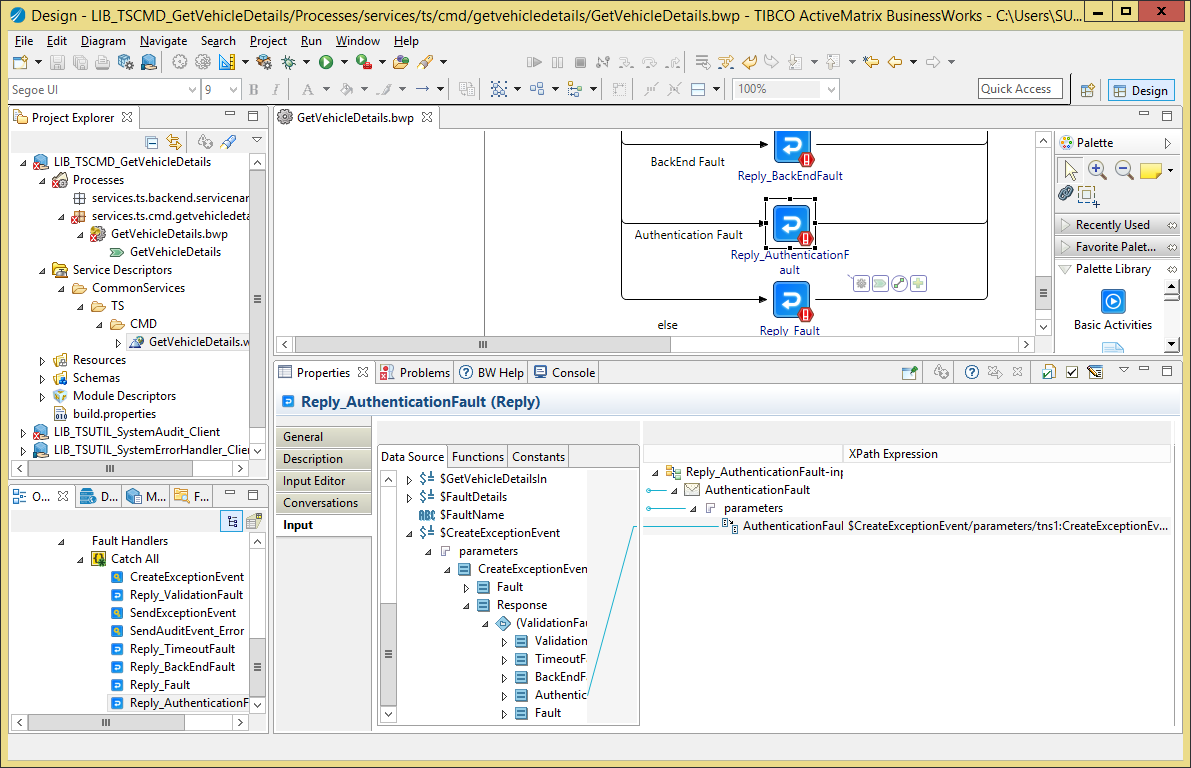


Figure 14: TS Service Provider Shared Module: Reply\_AuthenticationFault mapping

* + - 1. SendAuditEvent

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. 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.
   * + 1. 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. 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.
   * 1. Back End on HTTP/HTTPS

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

* + - 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.
   * + 1. Change the Module Properties
4. Go to the module properties.
5. 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>”.

***Example:***

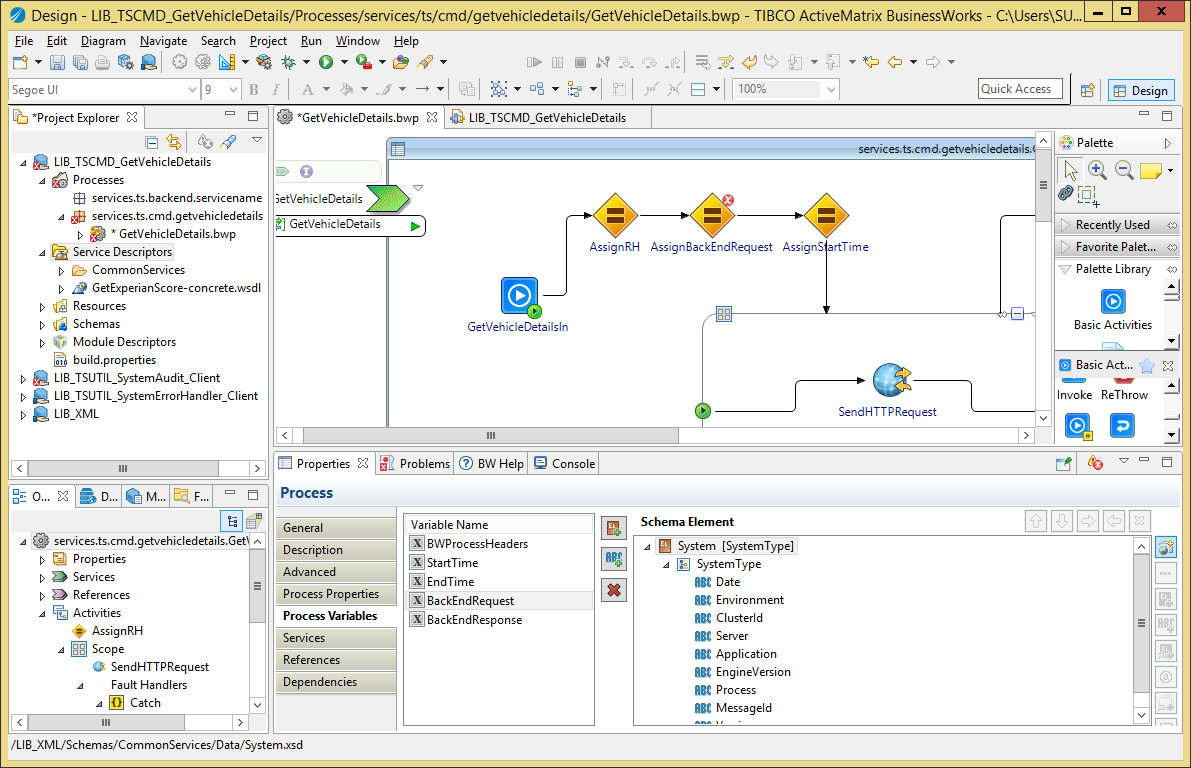
services.ts.<area>.<servicename> / <ServiceName> / backend / BACKEND becomes services.ts.bline.getcardscheme / GetCardScheme / backend / BLINE

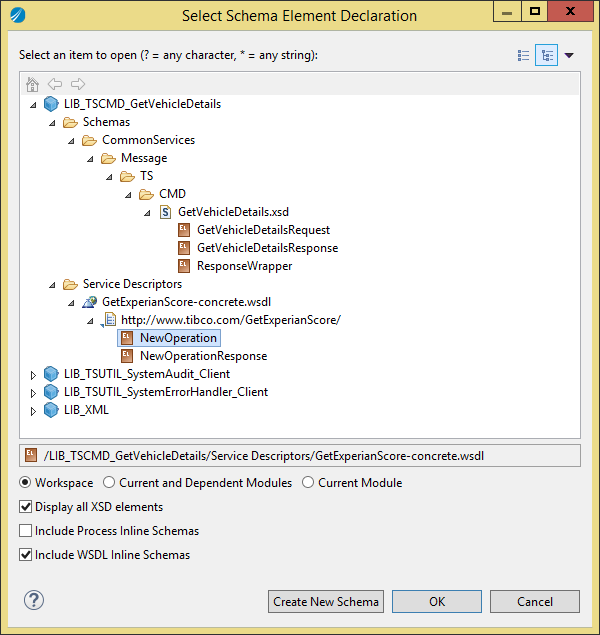
1. In the group you renamed, change the “endpoint.uri” value to correspond to the back-end URI.
2. Select the group “resources / services.ts.backend.servicename”. In the properties pane, use the light-bulb icon to rename the group into “services.ts.<area>.<servicename>”.
3. Select the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”. In the properties pane, use the light-bulb icon to rename the group into “Client-<BackEndName>”.
4. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-jdbc”.
   * + 1. Change Service Process (if back-end is accessed via SOAP)
5. 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:
   1. The HTTP client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.
   2. 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).



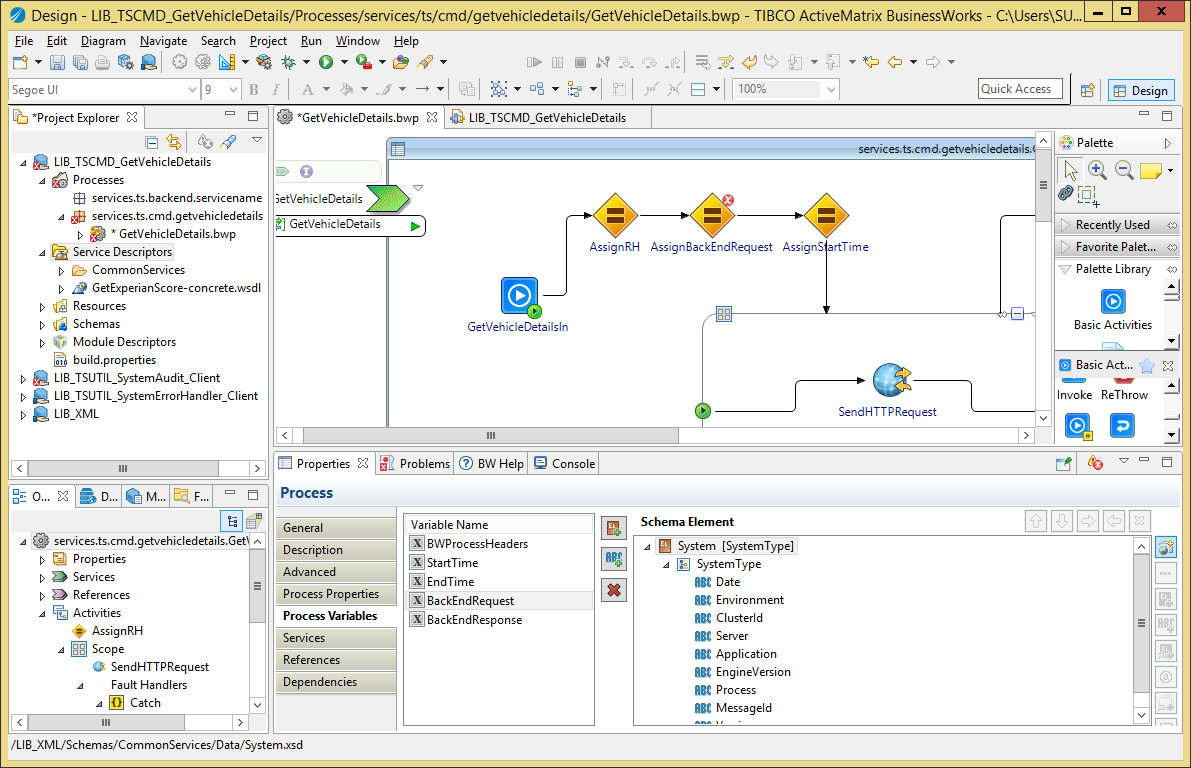


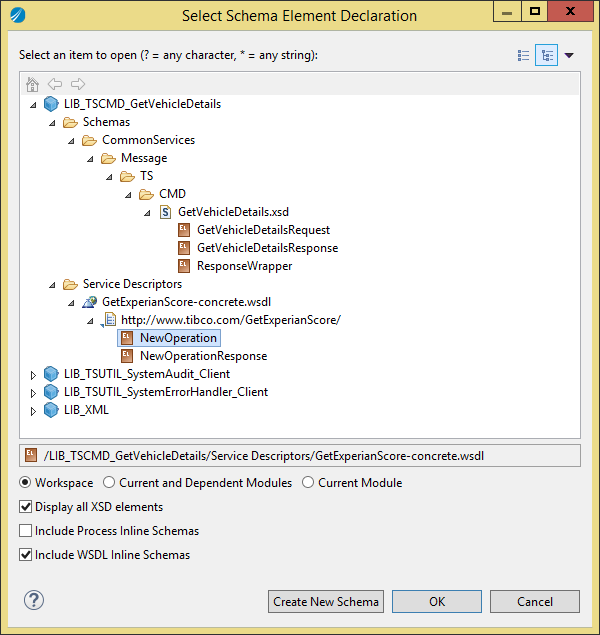
1. Edit the process variable “$BackEndResponse” so is uses the XML schema of the back-end operation response (from its concrete WSDL).
2. Map the request to the back-end in the “AssignBackEndRequest”. This task stores the request into the process variable $BackEndRequest so it can be re-used in other tasks (for the SystemAudit tasks for example).
3. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.
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).
   * + 1. 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.





1. Edit the process variable “$BackEndResponse” so is uses the element “Payload” from the **LIB\_XML / Schemas / CommonServices / Data / Message / AuditEvent.xsd** XSD.
2. 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).
3. Change the mapping of the Invoke task so it re-uses the $BackEndRequest variable.
4. 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).
   * + 1. 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\_BackEndFault” task and the transition from “AssignBackEndResponse” task to “Throw\_BackEndFault” task. In the “Throw\_BackEndFault” task you can change the text and description of the issue.

If not, you can delete the “Throw\_BackEndFault” task.

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

1. Right-click on the Invoke task and in the contextual menu, select the option “Catch / <fault>” where <fault> if the SOAP fault to handle.
2. In the new Catch block, add a ”Throw” task.
3. In “Input Editor” of the “Throw” task, add the XML element “ThrowableBackEndFault” from the “Schemas / CommonServices / Data / Message / Internal / Throwable” XSD.
4. 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.

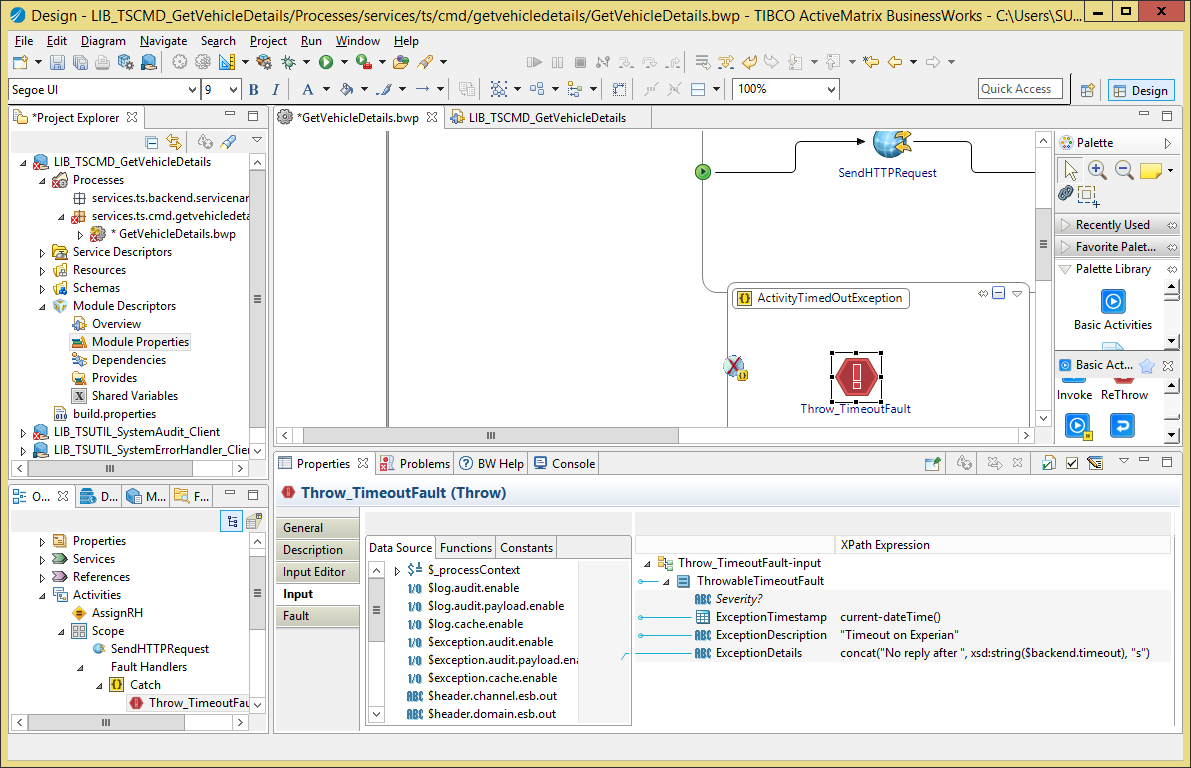


Figure 15: TS Service Provider Shared Module: Throw\_TimeoutFault mapping

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

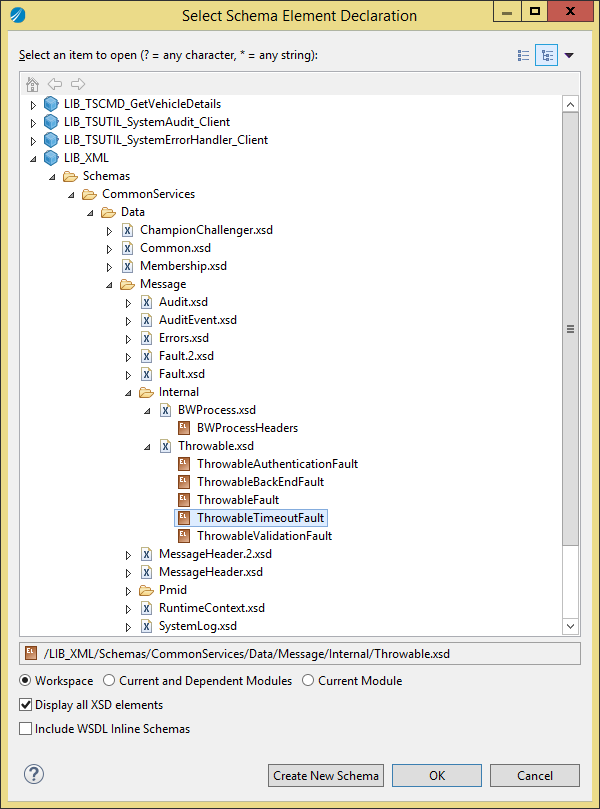


Figure 16: TS Service Provider Shared Module: Throw\_TimeoutFault configuration

1. Delete and recreate the task if necessary.
   * + 1. 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)
    1. Back End on JDBC

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

* + - 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.
   * + 1. Change the Module Properties
4. Go to the module properties.
5. 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>”.
6. In the group you renamed, delete the “endpoint.uri” property.
7. 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>”.
8. 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>”.
9. Delete the group “resources / services.ts.<area>.<servicename> / Client-BACKEND-http”.
   * + 1. 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:
   1. The JDBC client resource “services.ts.<area>.<servicename>.Client-<BackEndName>”.
   2. 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\_BackEndFault” task and the transition from “AssignBackEndResponse” task to “Throw\_BackEndFault” task. In the “Throw\_BackEndFault” task you can change the text and description of the issue.

If not, you can delete the “Throw\_BackEndFault” task.

1. In the “Throw\_TimeoutFault” task, you can change the text and description of the timeout error.
   1. Implement the Service
      1. Service Response

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

* **CorrelationId**: $BWProcessHeaders / CorrelationId
* **TransactionId**: $BWProcessHeaders / TransactionId
* **Version**: $BWProcessHeaders / Response / Version
* **ResponseDateTime**: current-dateTime()
* **Source**: copy of $BWProcessHeaders / Response / Source

TBC

* + 1. Invoking another ESB service

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

* **CorrelationId**: $BWProcessHeaders / CorrelationId
* **TransactionId**: $BWProcessHeaders / TransactionId
* **Version**: The version of the invoked service, for example “1.0”.
* **RequestDateTime**: current-dateTime()
* **Source**: copy of $BWProcessHeaders / Invoke / Source
  + 1. 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 3.4.6), but the “Throw\_ValidationFault” happens between the “AssignRH” task and “AssignBackEndRequest” task.

1. 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.

The JMS queue name is: AA.UK.Q.RQ.TS.<Area>.<ServiceName>.SOAP11.1.

* 1. Template Overview

The template module “APP\_ServiceTemplate.module” (renamed as APP\_BSServiceTemplate.module in version 0.4.3) 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.
  + **FlushExceptionEvents**: if ESB System Error Handler events are cached in memory, this process is triggered by a timer repeatedly to flush the cache into the ESB System Error Handler JMS destination.
  + **FlushLogEvents**: if ESB System Audit events are cached in memory, this process is triggered by a timer repeatedly to flush the cache into the ESB System Audit JMS destination.
* 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:***

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.
   1. Pre-Requisites

The XSD and WSDL have been created.

* + 1. Create the Application Module

With Windows Explorer:

1. Copy the Service Application Module template folder “APP\_TSServiceTemplate.module” into the folder **<SVN> / 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

1. Start TIBCO BusinessStudio
2. In your workspace, import the following modules:
   1. LIB\_XML (from <SVN> / trunk / XML
   2. LIB\_TSUTIL\_SystemAudit\_Client (from <SVN> / trunk / BW / TechnicalServices)
   3. LIB\_TSUTIL\_SystemErrorHandler\_Client (from <SVN> / trunk / BW / TechnicalServices)
   4. APP\_TS<Area>\_<ServiceName> (from <SVN> / trunk / BW / TechnicalServices)

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

1. Go to the module overview.
2. Change the name into “APP\_<TYPE><Area>\_<ServiceName> Module”:
   * 1. 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.

* + 1. Change the JMS Queue name

Edit the module properties as following:

1. Change the value of the property **services.ts.<area>.<servicename> / <ServiceName> / jms.queue** into AA.UK.Q.RQ.TS.<Area>.<ServiceName>.SOAP11.1.
   * 1. Change the JMS Connection
2. Rename the JNDI Client resource from “services.ts.backend.servicename.JNDIClient-ESB01” into “servies.ts.<area>.<sevicename>.JNDIClient-ESB01” (only the package changes).
3. Rename the JNDI Client resource from “services.ts.backend.servicename.JMSClient-ESB01” into “servies.ts.<area>.<sevicename>.JMSClient-ESB01” (only the package changes).
4. Delete the old resource package services.ts.backend.servicename.
   * 1. 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”:

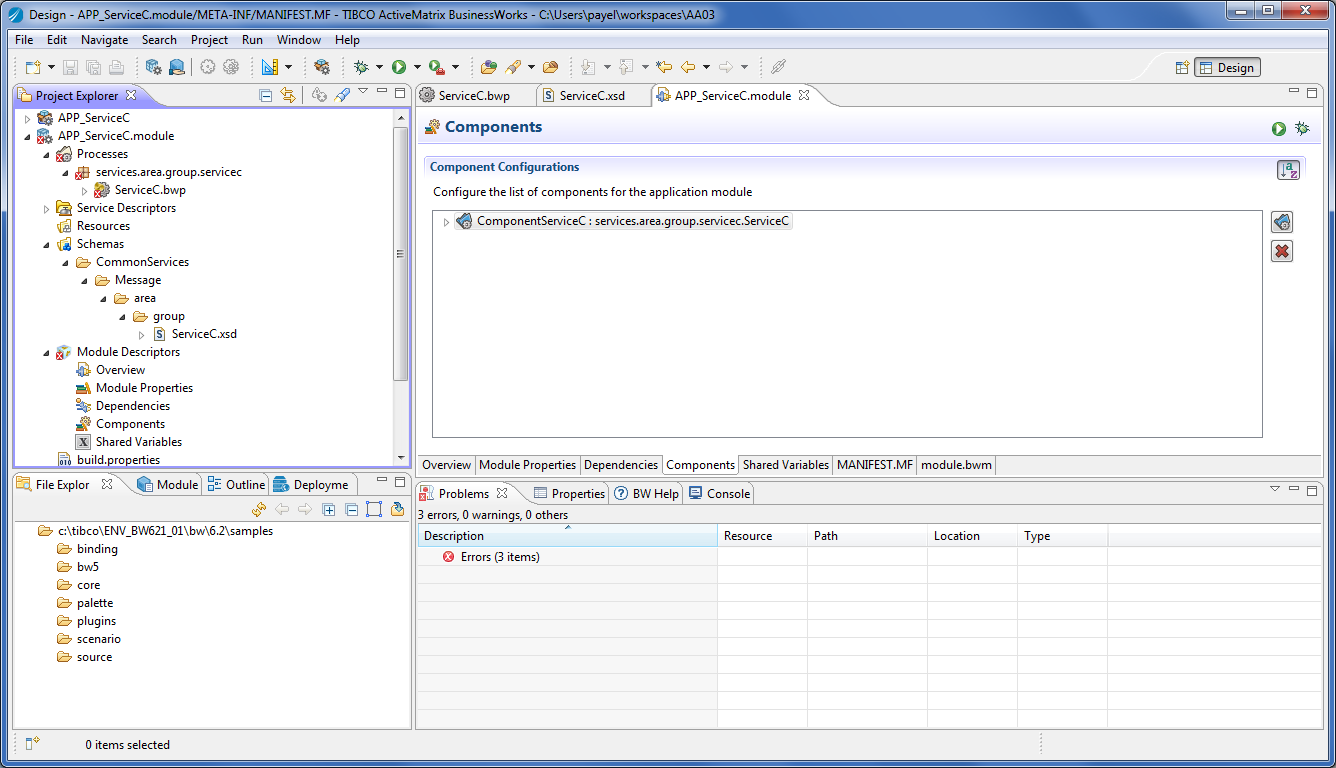


Figure 17: Create the SOAP over JMS Interface: Module Components

1. 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):

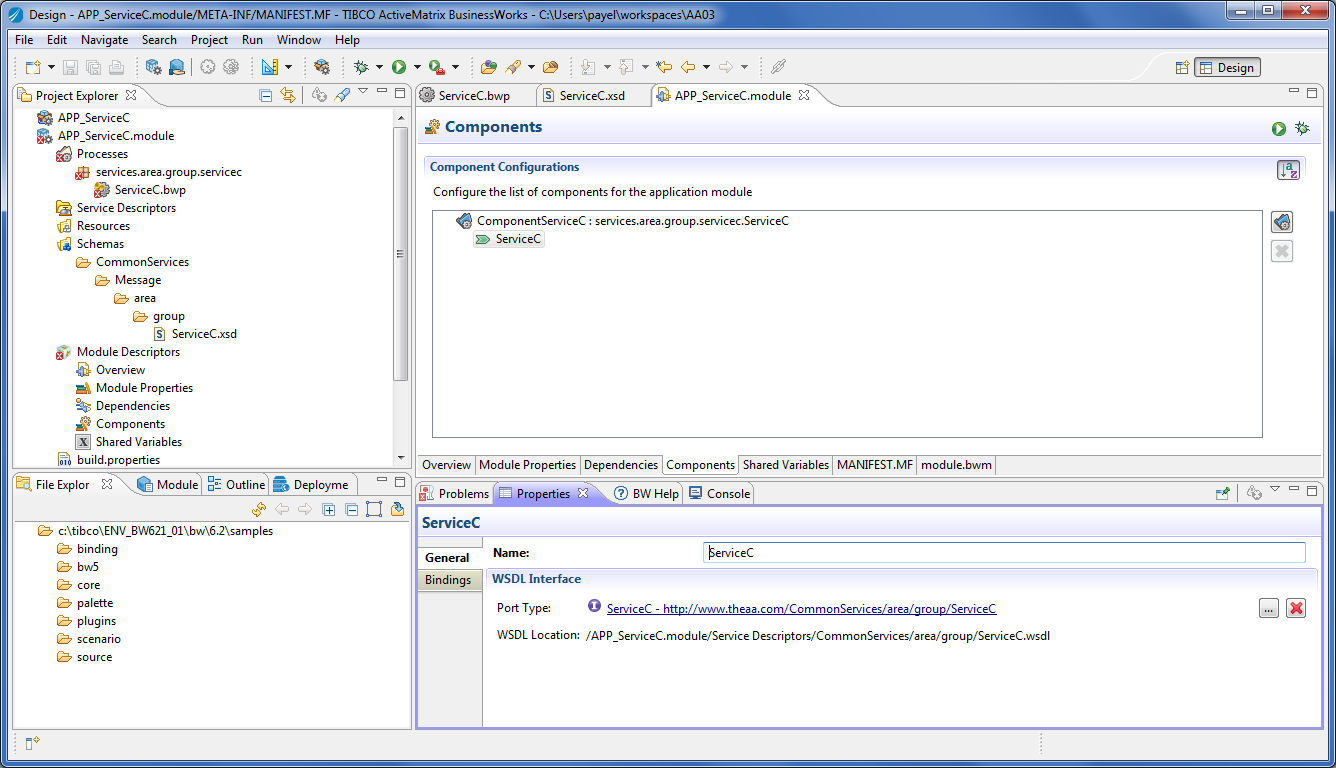


Figure 18: Create the SOAP over JMS Interface: Service Settings

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



Figure 19: Create the SOAP over JMS Interface: New SOAP Binding

1. Click on the “Finish” button.

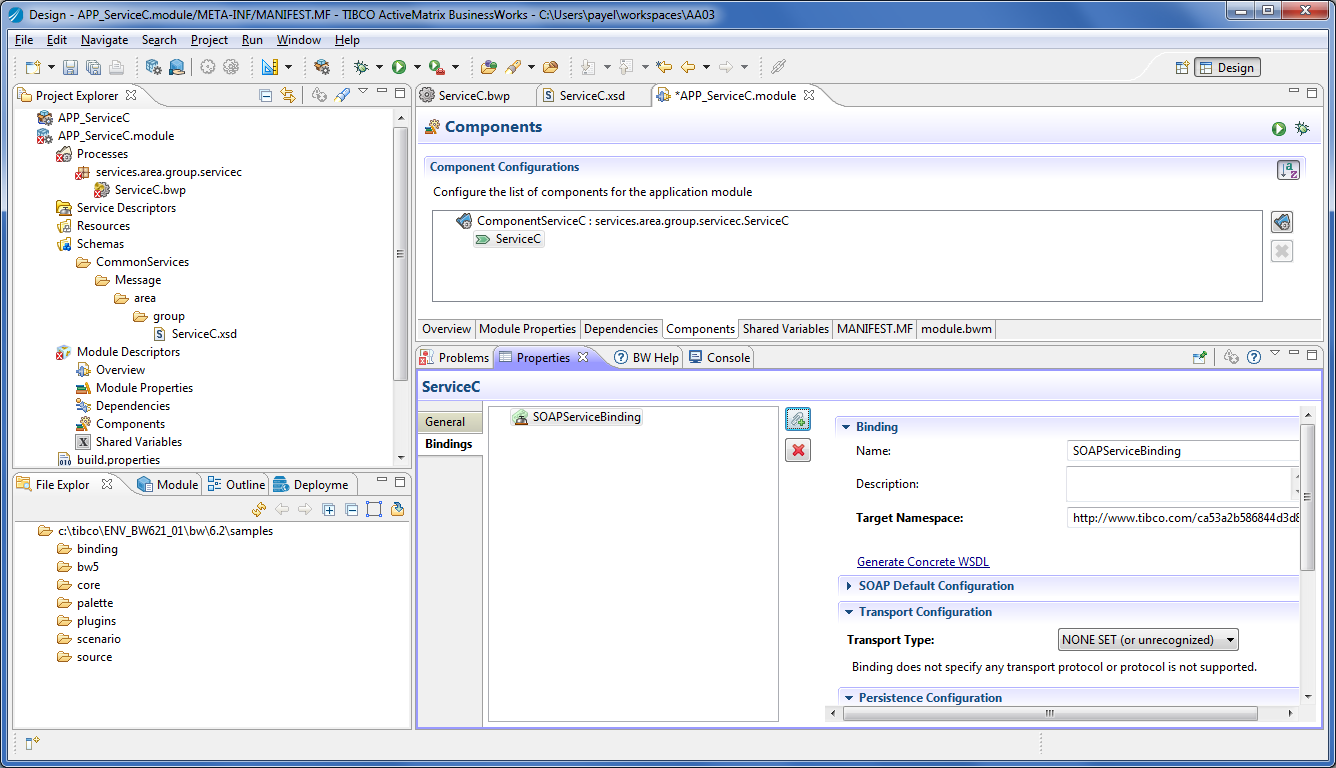


Figure 20: Create the SOAP over JMS Interface: New SOAP Binding

1. In the new “Binding” panel that appeared:
   1. Select “JMS” in the “Transport Type”.

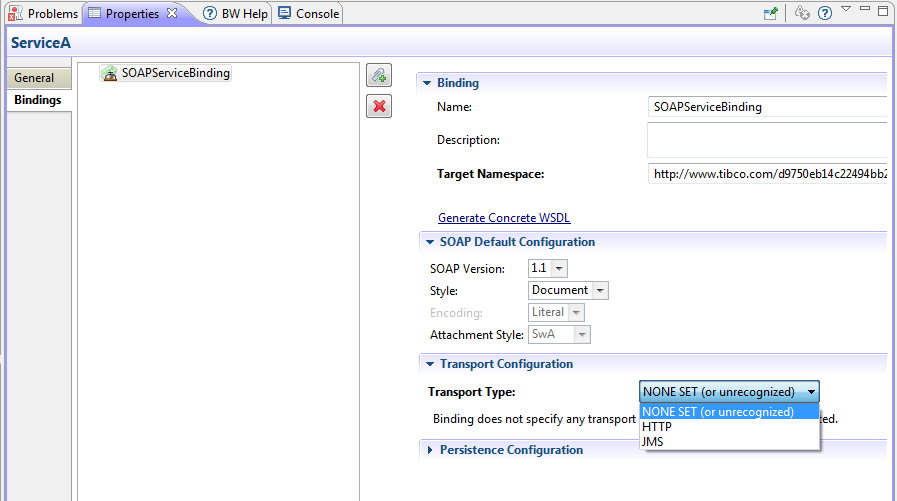


Figure 21: Create the SOAP over JMS Interface: Transport Type

1. In the “Select Jms… Dialog” box, select the JMS connection from the current module (the JMS Client resource renamed in previous steps)/
2. Now, back into the “Binding” panel:
   1. Select Acknowledgment Mode as “Auto”,
   2. Select JMS Message Type” as “Text”,
   3. Select Messaging Style as “Queue”.

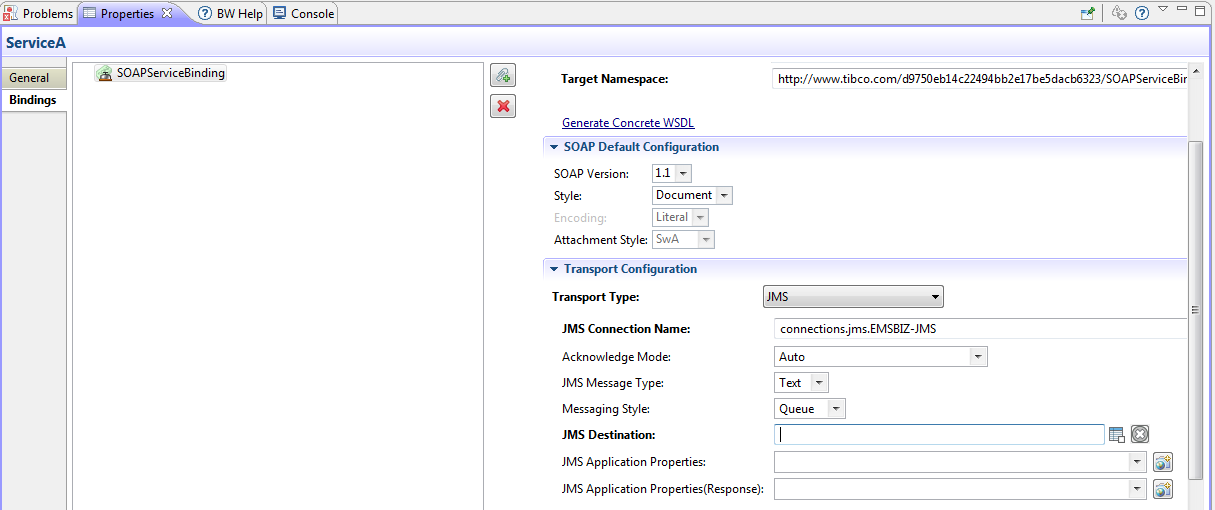


Figure 22: Create the SOAP over JMS Interface: JMS Settings

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

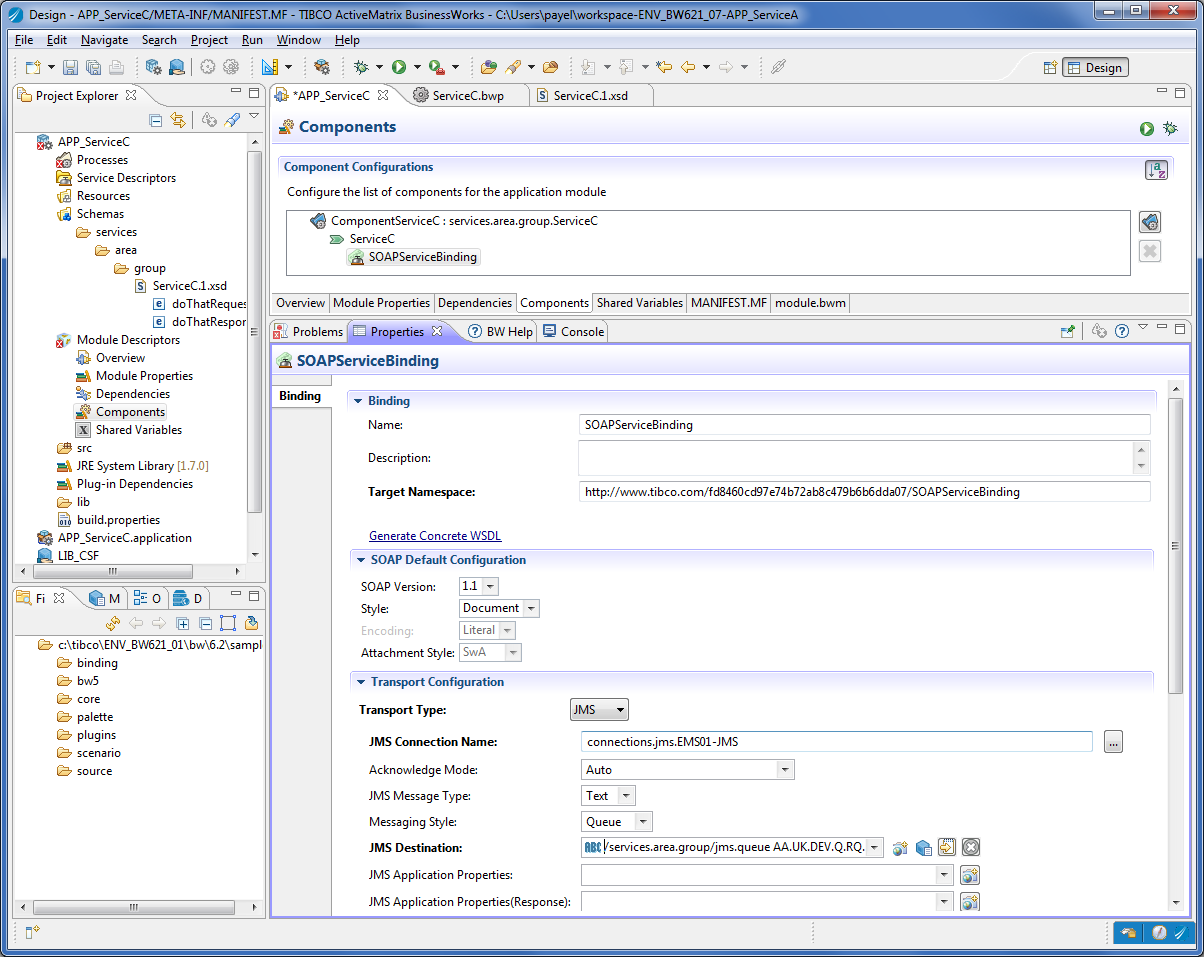


Figure 23: Create the SOAP over JMS Interface: Queue Name as Property

1. Save.
   * 1. 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 <SVN> / 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,overflowPolicy=discardOld

grant queue <queue> user=bw send,receive

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

commit

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

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

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

1. Replace <author>, <date>, <queue> by the appropriate values.
   * 1. 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 **<SVN> / trunk / EMS / rollback** folder create the text file “undeploy-TS<AR>\_<ServiceName>.ems” with the following content:
2. ###############################################################################
3. # This script un-deploys EMS resource for the service TS<AR>\_<ServiceName>
4. #
5. # Version 1.0.0 - <author> - <date>
6. #
7. # Changes:
8. # 1.0.0 - <date> - First version
9. #
10. ###############################################################################
11. ###############################################################################
12. # Queues
13. delete queue <queue>
14. ###############################################################################
15. commit
16. ###############################################################################
17. ### END OF FILE #############################################################
18. ###############################################################################
19. Replace <author>, <date>, <queue> by the appropriate values.

# Appendix

* 1. How to Upgrade to Version 0.4.4

This section explains how to upgrade existing service code to comply with template version 0.4.4.

* + 1. General

Process as following:

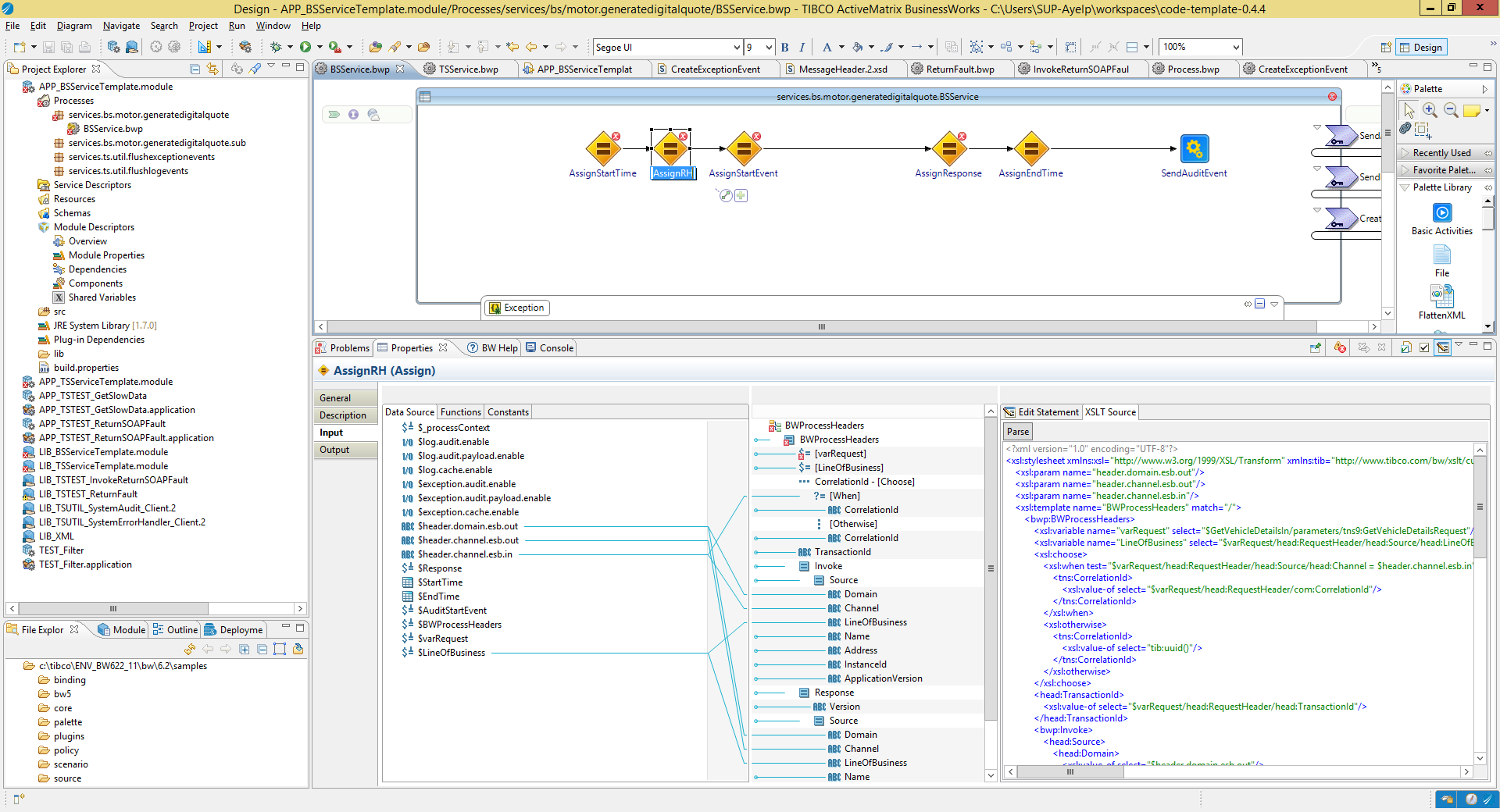
1. Update LIB\_XML / Schemas / CommonServices / Data / Message / MessageHeader.2.xsd with the new version contained in the new template.

In your existing modules:

1. Remove the dependency to LIB\_SystemAudit\_Client.
2. Remove the dependency on LIB\_SystemErrorHandler\_Client module.
3. Add a dependency on LIB\_SystemAudit\_Client.2 module.
4. Add a dependency on LIB\_SystemErrorHandler\_Client.2 module.
   * 1. Business Services

Modify the service process as following:

1. Import the service template APP\_BSServiceTemplate.module into your workspace.
2. In the module, goes to the BSService.bwp process and select the “AssignRH” task.
3. Go to the input mapping.
4. Right-click on any element on the right panel and select “Show Edit Tab” to you can see the mapping XSLT code:



1. Select all the code and type Ctrl-C to copy it into the Windows Clipboard.
2. Go to your service, select the service process and “AssignRH task”.
3. Go to the input mapping.
4. Right-click on any element on the right panel and select “Show Edit Tab” to you can see the mapping XSLT code.
5. Select all the code and type Ctrl-V to paste the code copied from the template.
6. Click on the “Parse” button so the mapping get refreshed.
7. Fix the mapping of the “varRequest” as in section 3.3.6.1.
8. In the task “AssignRH\_InputError”, change all values of LineOfBusiness into “GEN”.

What did change;

* There is a new variable LineOfBusiness (after “varRequest”) that stores the LineOfBusiness from the request message.
* The **Invoke / Source / LineOfBusiness** and **Response / Source / LineOfBusiness** elements are now mapped to this new variable.
  + 1. Technical Services

Modify the service process as following:

1. Import the service template APP\_TSServiceTemplate.module into your workspace.
2. In the module, goes to the TSService.bwp process and select the “AssignRH” task.
3. Go to the input mapping.
4. Right-click on any element on the right panel and select “Show Edit Tab” to you can see the mapping XSLT code.
5. Select all the code and type Ctrl-C to copy it into the Windows Clipboard.
6. Go to your service, select the service process and “AssignRH task”.
7. Go to the input mapping.
8. Right-click on any element on the right panel and select “Show Edit Tab” to you can see the mapping XSLT code.
9. Select all the code and type Ctrl-V to paste the code copied from the template.
10. Click on the “Parse” button so the mapping get refreshed.
11. Fix the mapping of the “varRequest” as in section 3.3.6.1.

What did change;

* There is a new variable LineOfBusiness (after “varRequest”) that stores the LineOfBusiness from the request message.
* The **Invoke / Source / LineOfBusiness** and **Response / Source / LineOfBusiness** elements are now mapped to this new variable.