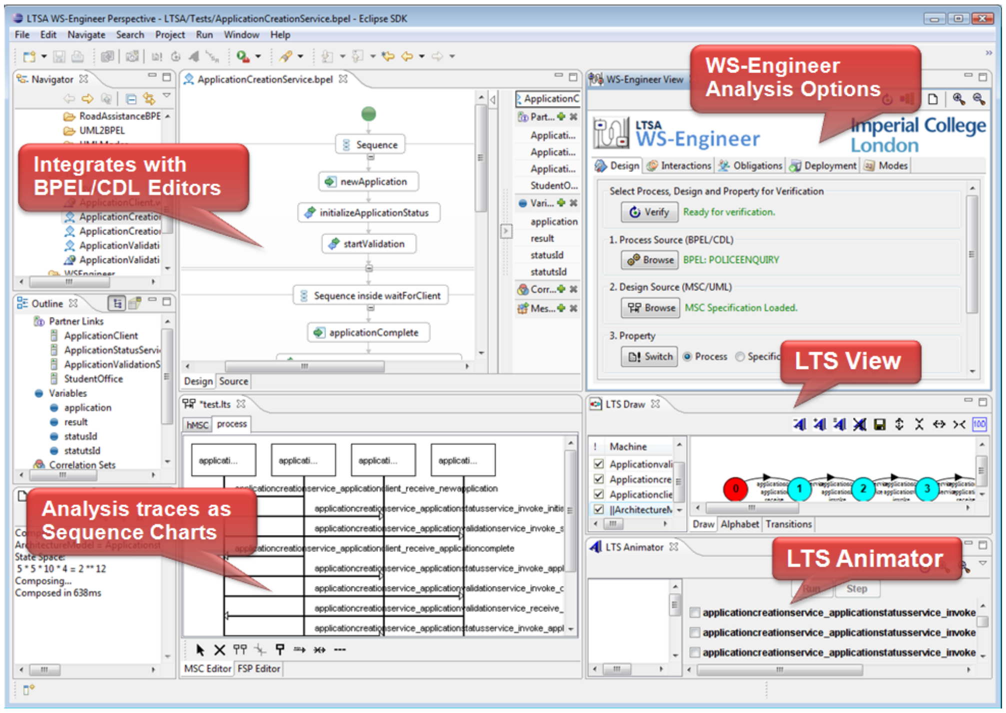
Imperial College London

Department of Computing

The Labelled Transition System Analyser

LTSA WS-Engineer Help Guide

A Plug-in for LTSA Eclipse



WS-Engineer Analysis Options
Integrates with BPEL/CDL Editors
LTS View
Analysis traces as Sequence Charts
LTS Animator

© 2009 Imperial College London

Imperial College London

Department of Computing The Labelled Transition System Analyser

LTSA WS-Engineer Help Guide

Publisher

Imperial College London

Managing Editor

Dr Howard Foster

Technical Editors

Dr Howard Foster

Special thanks to:

Special thanks are given to the following people who contributed to the WS-Engineer environment, tool design and mechanisms.

Professor Jeff Magee, Department of Computing, Imperial College London

Professor Jeff Kramer, Department of Computing, Imperial College London

Dr Sebastian Uchitel, Department of Computing, Imperial College London

Table of Contents

1 Getting Started

1.1 Introduction

WS-Engineer Plug-in For LTSA Eclipse

The LTSA WS-Engineer plug-in is an extension to the LTSA Eclipse Plug-in which allows service models to be described by translation of the service process descriptions, and can be used to perform model-based engineering including:

- Support for BPEL4WS 1.1, WS-CDL 1.0 and WSIF (Web Service Invocation Framework) Log analysis.
- Analysis of interaction models for specified properties (e.g. trace equivalence, deadlock, liveness etc)
- Synthesis of composition and choreography interaction processes from behaviour models
- Verfication of service behaviour models (e.g. for partner obligations)
- Validation through interactive and animated models
- Trace analysis for service composition interaction logs (e.g. BPWS4J Logs)

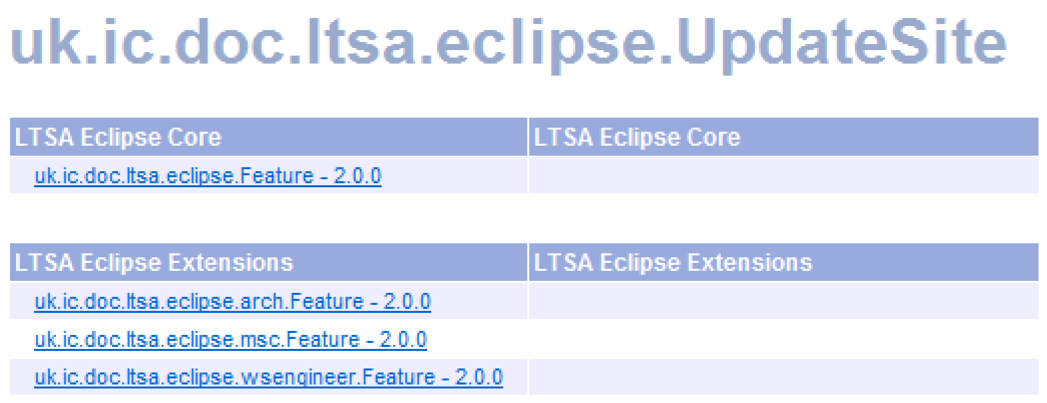
The WS-Engineer Tool is based upon the work by Dr. Howard Foster in his PhD thesis: A Rigorous Approach to Engineering Web Service Compositions.

1.2 Installation

This plug-in requires the LTSA Eclipse plug-in, and can be installed as a feature from the main LTSA Eclipse Update site.

Note: The install site for LTSA Eclipse is: http://www.doc.ic.ac.uk/ltsa/eclipse/install.

- Please see the LTSA Eclipse help guide for details of installing the core LTSA Eclipse and associated plug-ins (including WS-Engineer).
- Additionally you may want to install some specification language tool suites.
- There are also some Sample Files which can be used to perform analysis tutorials.



uk.ic.doc.ltsa.eclipse.UpdateSite

LTSA Eclipse Core	LTSA Eclipse Core
uk.ic.doc.ltsa.eclipse.Feature - 2.0.0	

LTSA Eclipse Extensions	LTSA Eclipse Extensions
uk.ic.doc.ltsa.eclipse.arch.Feature - 2.0.0	
uk.ic.doc.ltsa.eclipse.msc.Feature - 2.0.0	
uk.ic.doc.ltsa.eclipse.wsengineer.Feature - 2.0.0	

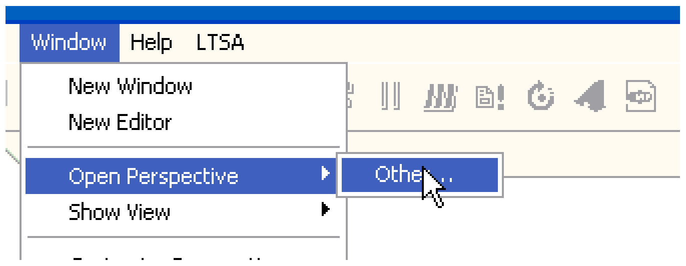
2 The WS-Engineer Environment

This section describes the WS-Engineer Perspective, WS-Engineer Views and the WS-Engineer Preferences Page.

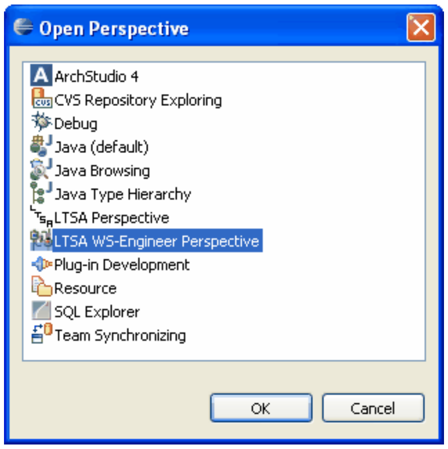
2.1 The WS-Engineer Perspective

WS-Engineer provides a pre-configured perspective with suggested views and layout set for use. To enable the perspective follow the steps below.

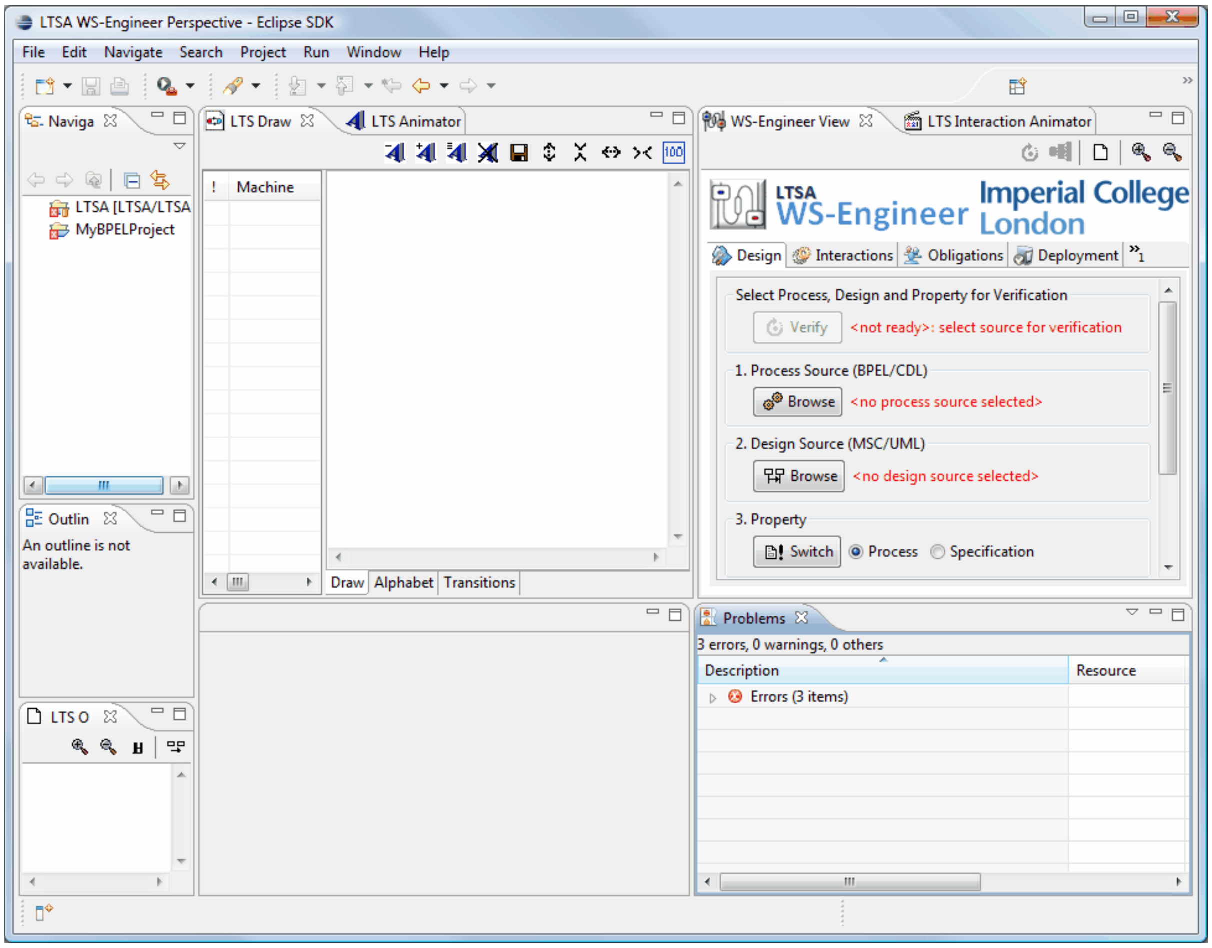
1. Locate the Eclipse menu bar. Select **Window -> Open Perspective -> Other**



2. Select **LTSA WS-Engineer Perspective** and click **OK.** If this menu item is not present, then the plug-in has not been successfully installed.



3. Notice the layout of views-windows will be similar to that illustrated in the figure below.

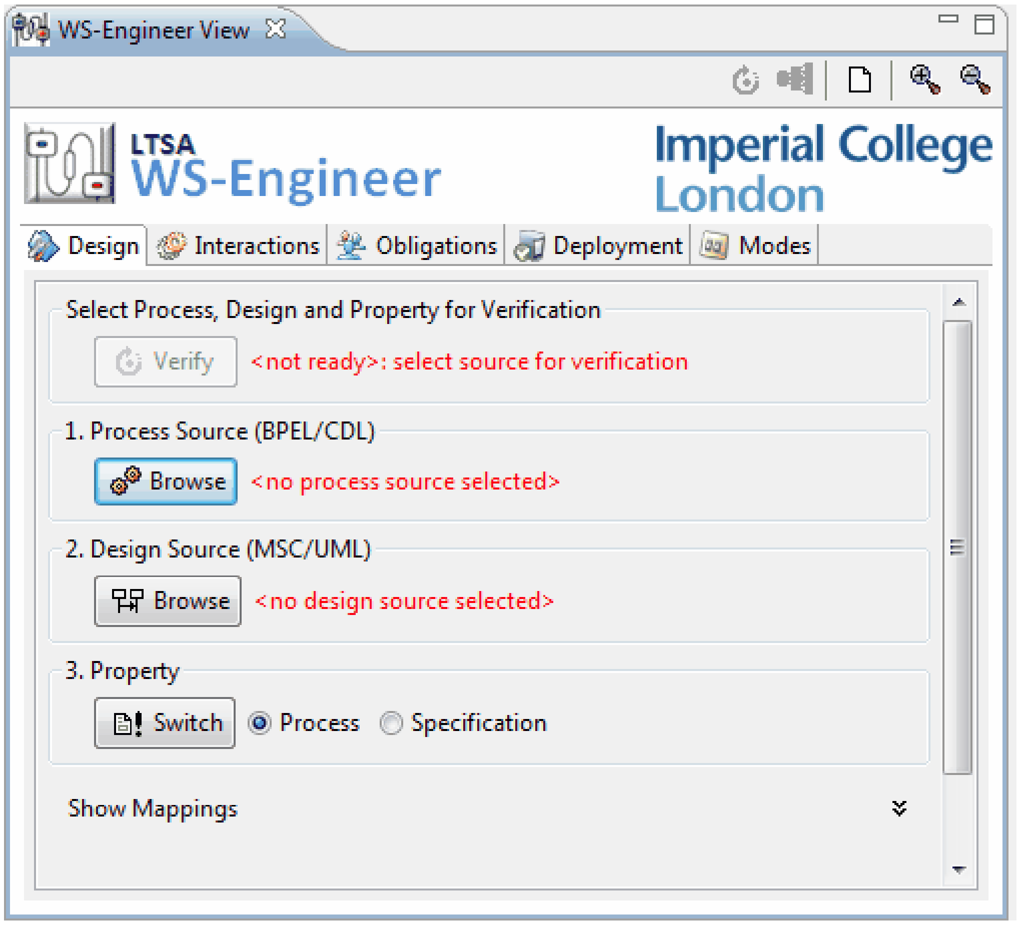

LTSA WS-Engineer Perspective - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Naviga
LTSA [LTSA/LTSA
MyBPELProject
LTS Draw
LTS Animator
Machine
Draw Alphabet Transitions
WS-Engineer View
LTS Interaction Animator
LTSA WS-Engineer
Imperial College London
Design Interactions Obligations Deployment
Select Process, Design and Property for Verification
Verify
<not ready>: select source for verification
1. Process Source (BPEL/CDL)
Browse
<no process source selected>
2. Design Source (MSC/UML)
Browse
<no design source selected>
3. Property
Switch
Process
Specification
Outlin
An outline is not available.
LTS O
Problems
3 errors, 0 warnings, 0 others
Description
Resource
Errors (3 items)

\

2.2 The WS-Engineer View

The WS-Engineer view presents the analysis features of the LTSA WS-Engineer tool in a series of tab pages.

Note: If this view is not visible, enable it by **Window** → **Show View** → **Other** → **LTSA WS-Engineer** → **WS-Engineer View**.



Select Process, Design and Property for Verification

Verify <not ready>: select source for verification

1. Process Source (BPEL/CDL)

Browse <no process source selected>

2. Design Source (MSC/UML)

Browse <no design source selected>

3. Property

Switch [✓ Process] Specification

Show Mappings

The design view can be accessed by selecting the Design tab on the WS-Engineer view tab list.

The interactions view can be accessed by selecting the Interactions tab on the WS-Engineer view tab list.

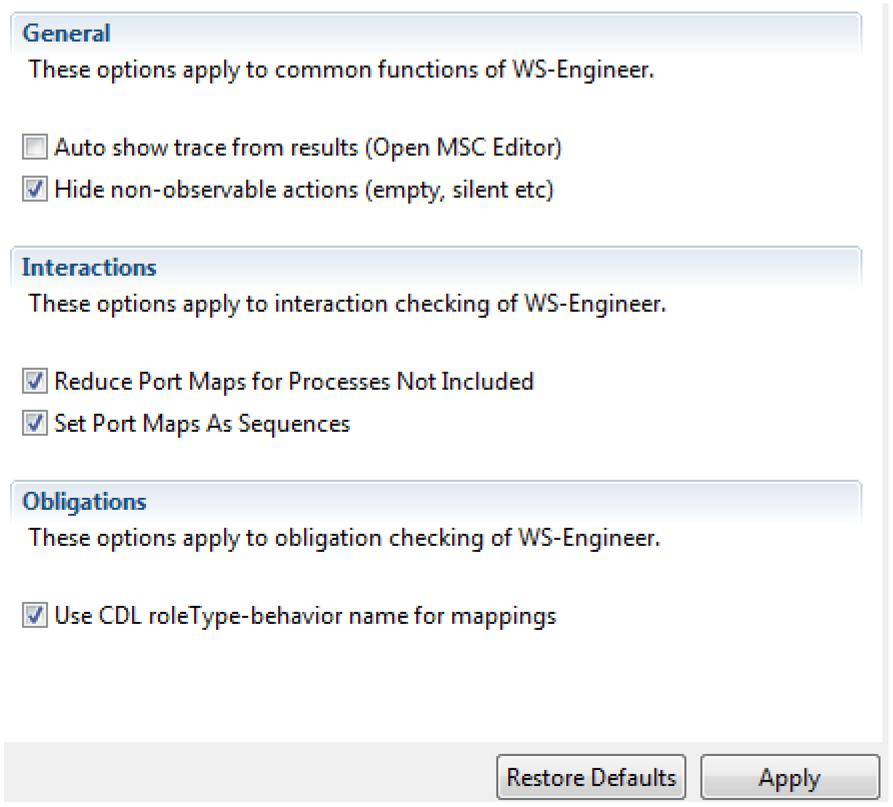
The obligations view can be accessed by selecting the Obligations tab on the WS-Engineer view tab list.

The deployment view can be accessed by selecting the Deployment tab on the WS-Engineer view tab list.

The modes view can be accessed by selecting the Modes tab on the WS-Engineer view tab list.

2.3 WS-Engineer Preferences

The WS-Engineer preferences page is accessed from the Eclipse menu item: **Window → Preferences → LTSA WS-Engineer.**



General
These options apply to common functions of WS-Engineer.

- [ ] Auto show trace from results (Open MSC Editor)
- [✓] Hide non-observable actions (empty, silent etc)

Interactions
These options apply to interaction checking of WS-Engineer.

- [✓] Reduce Port Maps for Processes Not Included
- [✓] Set Port Maps As Sequences

Obligations
These options apply to obligation checking of WS-Engineer.

- [✓] Use CDL roleType-behavior name for mappings

Further details of preferences can be found in the Preferences section of this help guide.

3 Tutorials

This section details some tutorials for using the WS-Engineer plug-in.

3.1 Working with WS-Engineer

WS-Engineer supports a number of WS-* standards and supporting tool sets.

Tool Sets

To make working with WS-Engineer easier, it is recommended to have one or more of the following tools.

View	Specification Language	Tools	Home Page	Eclipse Update Site (if available)
Design, Interactions, Obligations and	WS-BPEL 2.0	Eclipse BPEL Editor	www.eclipse.org/bpel	download.eclipse.org/technology/bpel/

View	Specification Language	Tools	Home Page	Eclipse Update Site (if available)
Deployment				update-site/
Obligations	WS-CDL 1.0	Pi4SOA WS-CDL Tool Suite	sourceforge.net/ projects/pi4soa/	
Deployment, Modes	xADL 2.0	ArchStudio	www.isr.uci.edu/ projects/archstudio/	www.isr.uci.edu/ projects/archstudio-update/
Deployment, Modes	UML 2.0	IBM Rational Software Architect	www-01.ibm.com/ software/awdtools/ architect/ swarchitect/	

Sample Files

There are Sample Files which can be used as source files described in this tutorial.

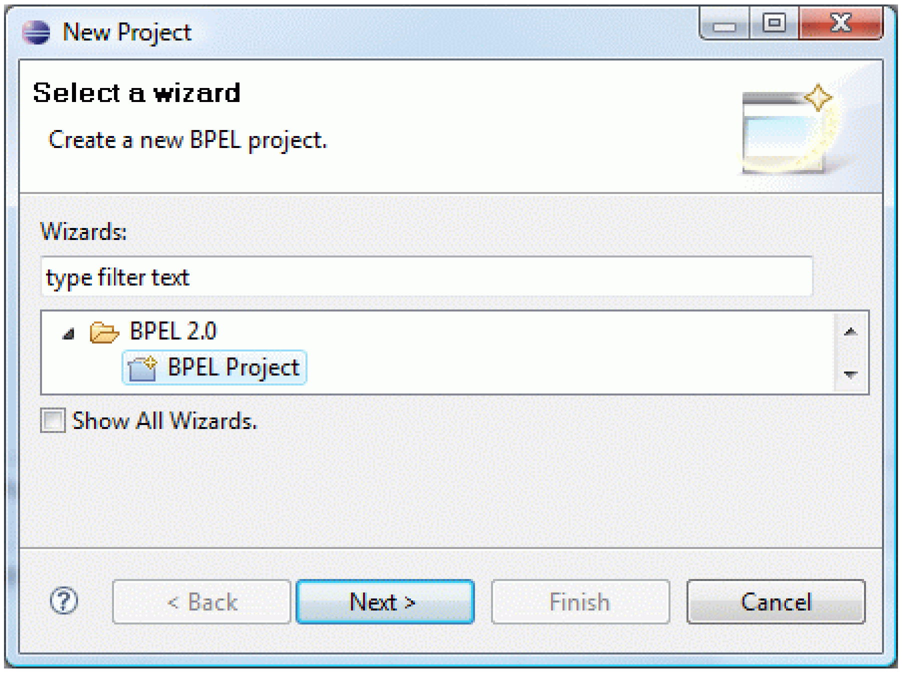
3.1.1 WS-BPEL and Eclipse BPEL Editor

In this tutorial the user will be able to create a new WS-BPEL process and validate the process through WS-Engineer LTSA Animator.

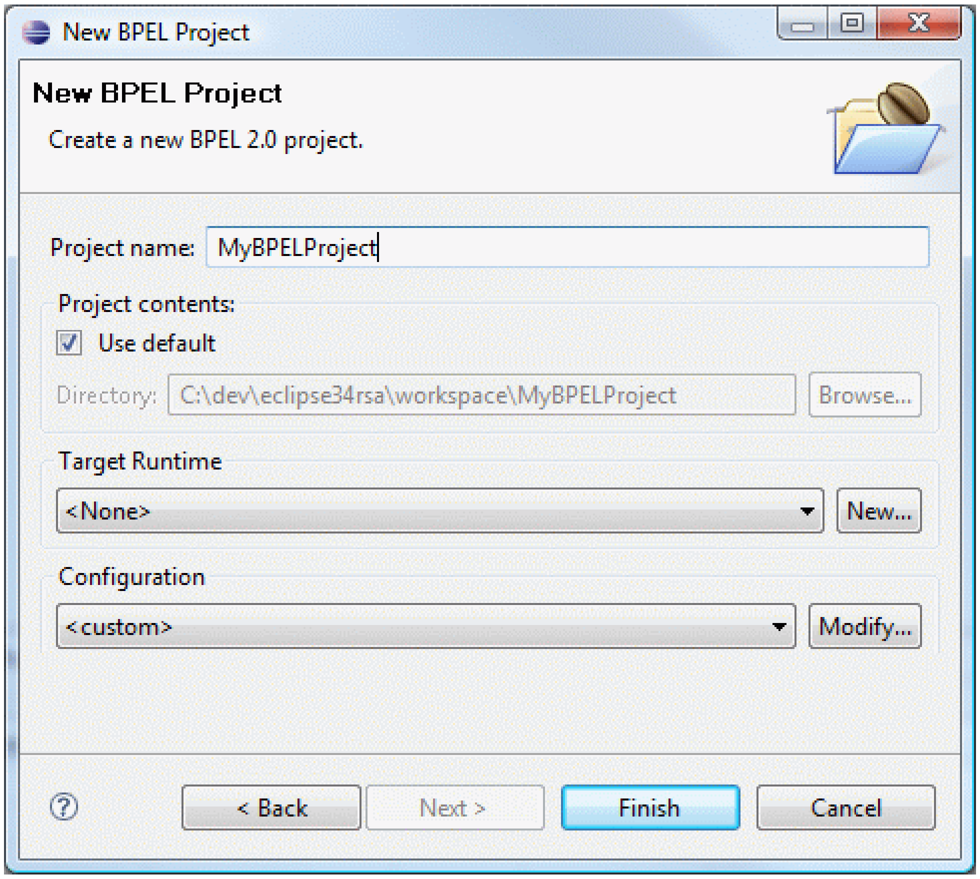
Note: This tutorial requires the use of the Eclipse BPEL Editor.

1. **Create a new BPEL project and BPEL process.**

a. From the Eclipse Menus select **File** → **New** → **Project** → **BPEL 2.0** → **BPEL Project** and select **Next.**



b. Name the project (e.g. MyBPELProject) and click **Finish.**



Project name: MyBPELProject

Project contents:

☑ Use default

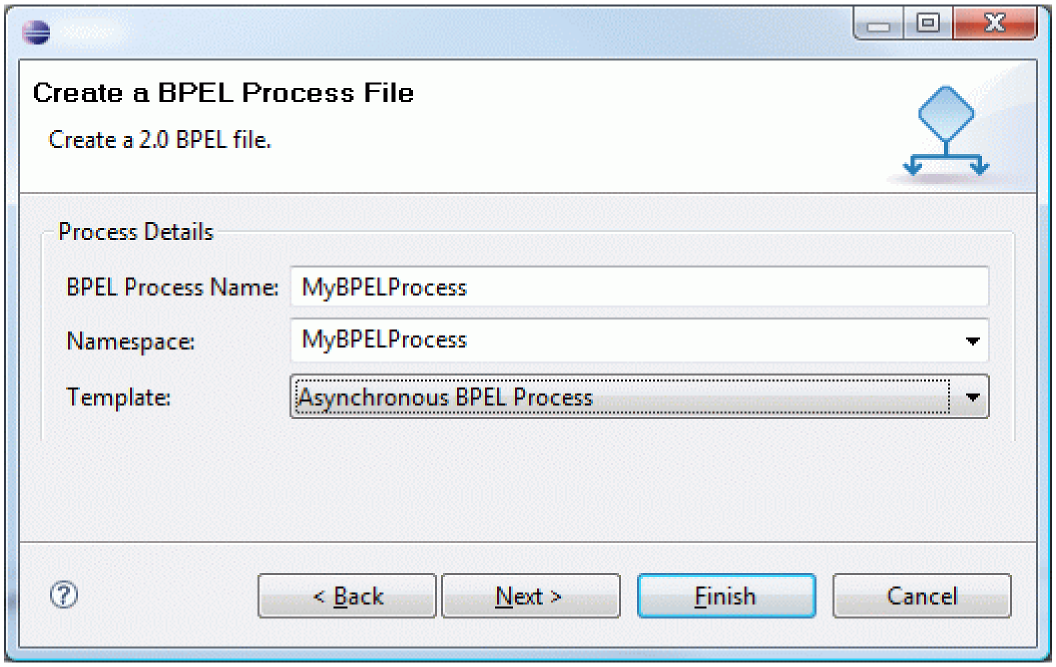
Directory: C:\dev\eclipse34rsa\workspace\MyBPELProject

Target Runtime: <None>

Configuration: <custom>

c. Select the new **MyBPELProject** project in the Package Explorer View.

d. Right-click and select **New → Other → BPEL 2.0 → New BPEL Process File** and select **Next.**
e. Enter a process name (e.g. MyBPELProcess), a namespace and keep the template as **Asynchronous BPEL Process**. Click **Finish**.


Create a BPEL Process File

Create a 2.0 BPEL file.

Process Details

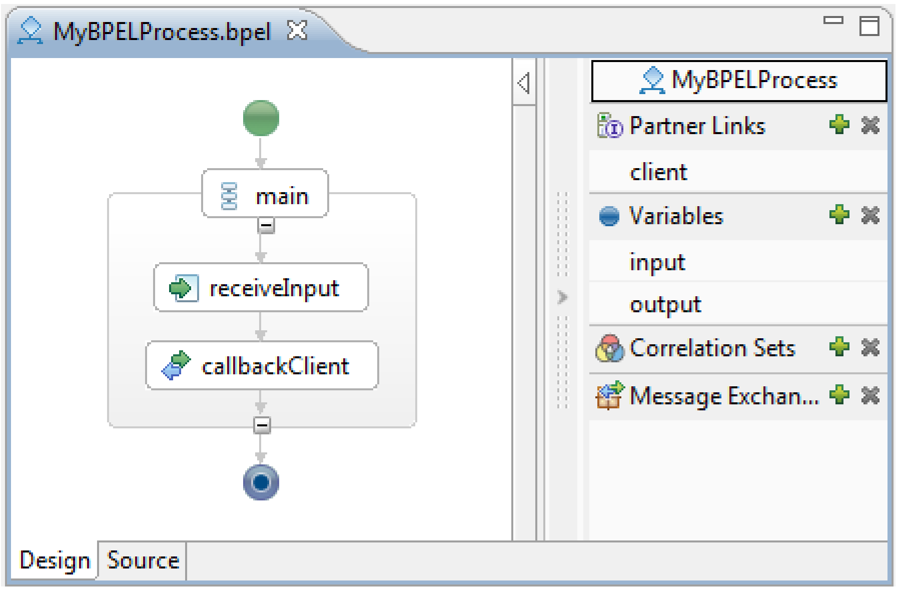
BPEL Process Name: MyBPELProcess

Namespace: MyBPELProcess

Template: Asynchronous BPEL Process

2. Open the new BPEL Process using the Eclipse BPEL Editor

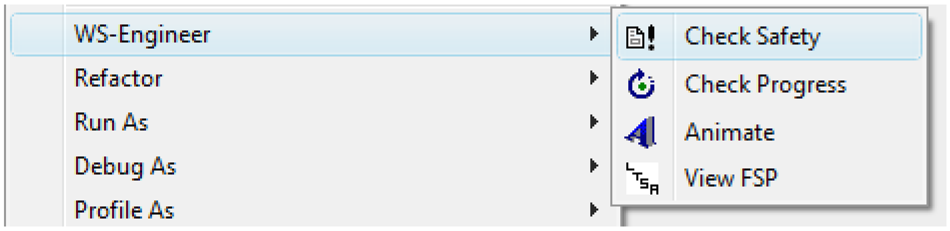
Note that two activities are included, one receiveInput (to receive a new request and create a new instance of the process) and callbackClient which invokes the reply.

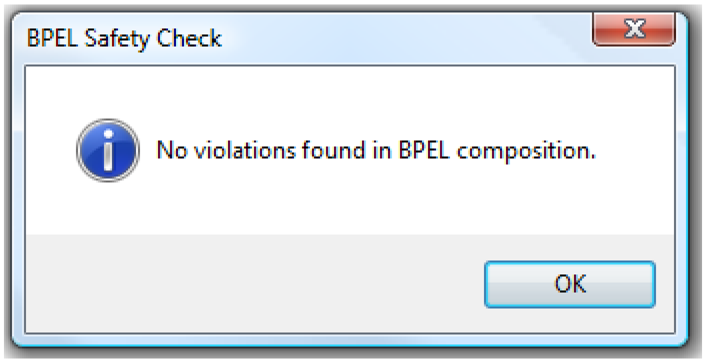


3. Check process "Safety"

A Safety check provides a default deadlock analysis through the process states.

a. Select a blank area in the process editor view and right-click.
b. Select **WS-Engineer → Safety Check.**



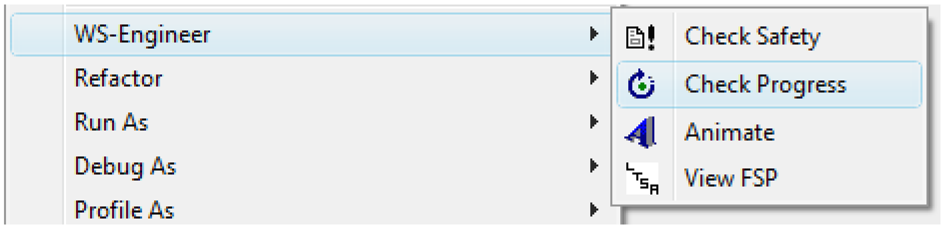


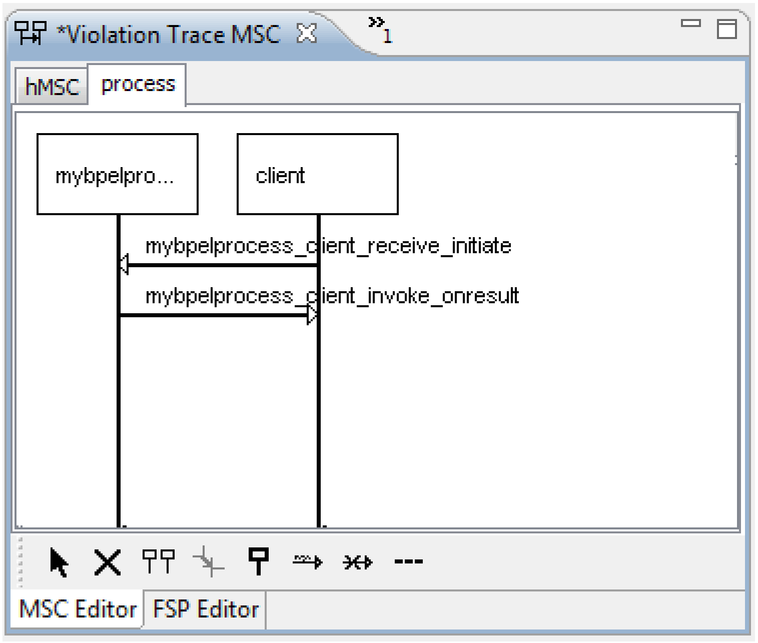
No violations found in BPEL composition.

4. Check process "Progress"

A Progress check provides a default trace through the process states.

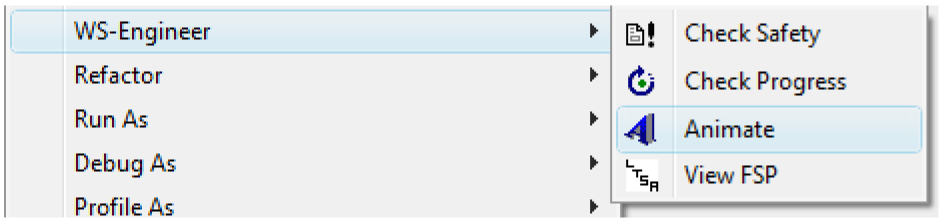
a. Select a blank area in the process editor view and right-click.
b. Select **WS-Engineer → Progress Check.** A view of the process as an MSC trace will be displayed**.**

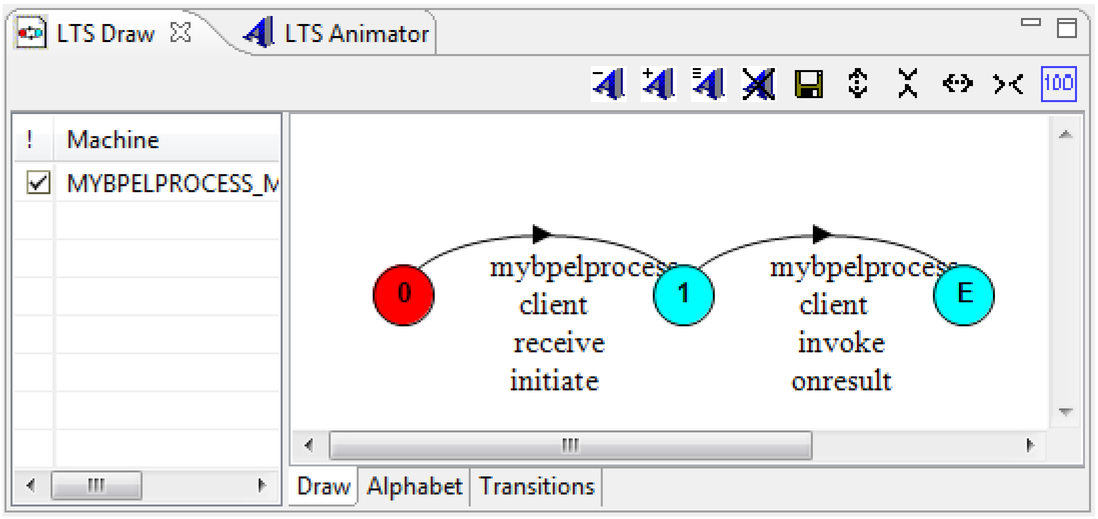



*Violation Trace MSC
hMSC
process
mybpelpro...
client
mybpelprocess_client_receive_initiate
mybpelprocess_client_invoke_onresult
MSC Editor
FSP Editor

5. View the Graphical LTS of the Process

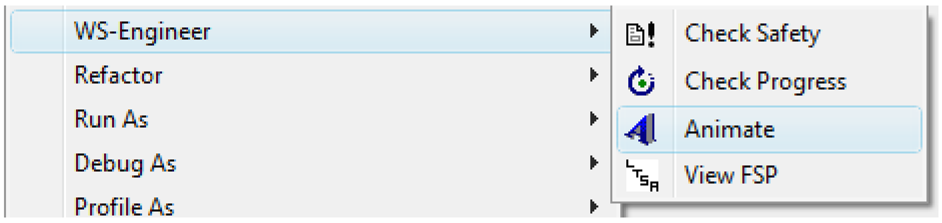
a. Select a blank area in the process editor view and right-click.
b. Select **WS-Engineer → Animate.**
c. Switch the LTS Draw view and you can see the state machine of the main process.

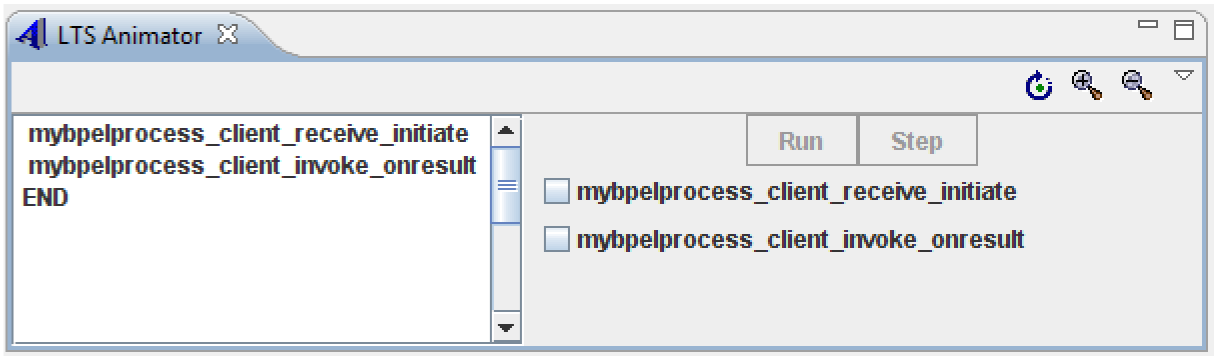




6. Animate the BPEL process

a. Select a blank area in the process editor view and right-click.
b. Select **WS-Engineer → Animate.**
c. Switch the LTS Animator view and you can step through the process state machine.





mybpelprocess_client_receive_initiate
mybpelprocess_client_invoke_onresult
END

mybpelprocess_client_receive_initiate
mybpelprocess_client_invoke_onresult

END OF TUTORIAL

3.1.2 WS-BPEL and LTSA-BPEL Editor

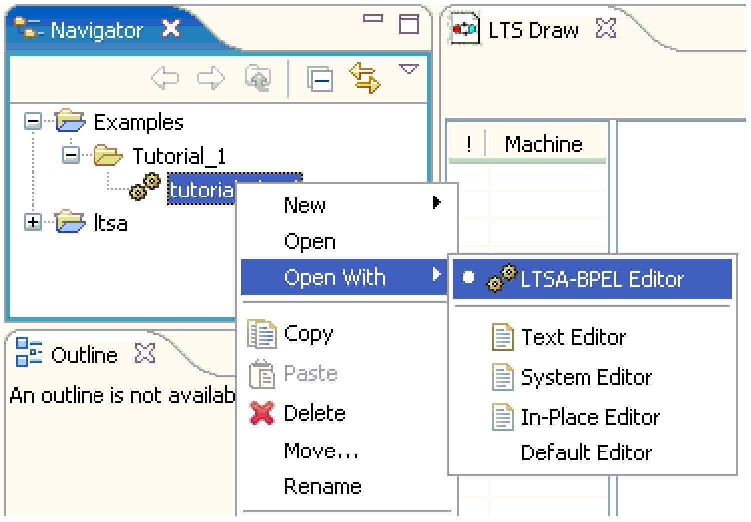
In this tutorial the user will be able to import a WS-BPEL process and generate a Finite State Process composition from the process.

1. Import BPEL4WS files in to a project

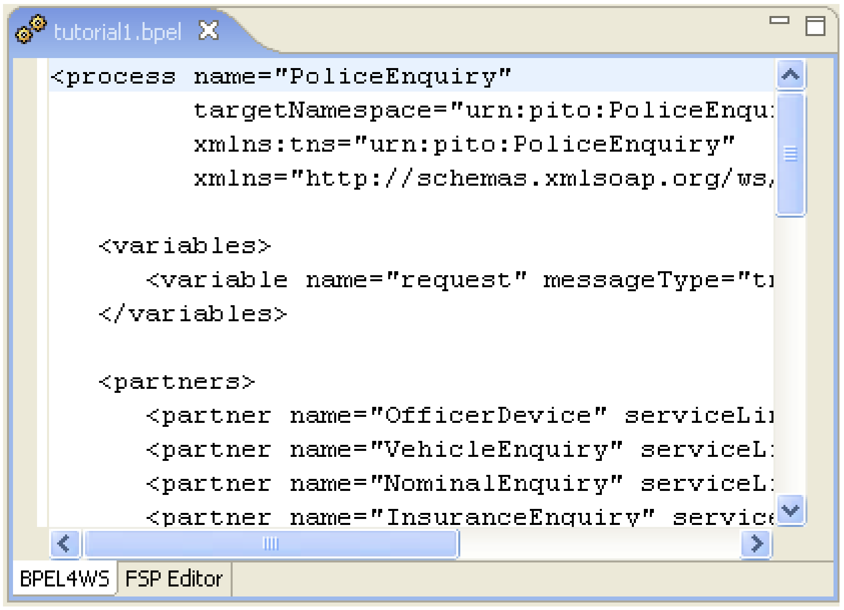
a. Locate your WS-BPEL source files and import them in to the project created previously.
b. In this example we use a WS-BPEL process file called "**tutorial1.bpel**".
c. You can substitute the name of your WS-BPEL files for this one at each step.

2. Open a WS-BPEL file in WS-Engineer

a. Locate a WS-BPEL file (with extension .bpel) in the Navigator view.
b. Select the file, and right-click the mouse, select **Open With → LTSA-BPEL Editor,** as illustrated below.

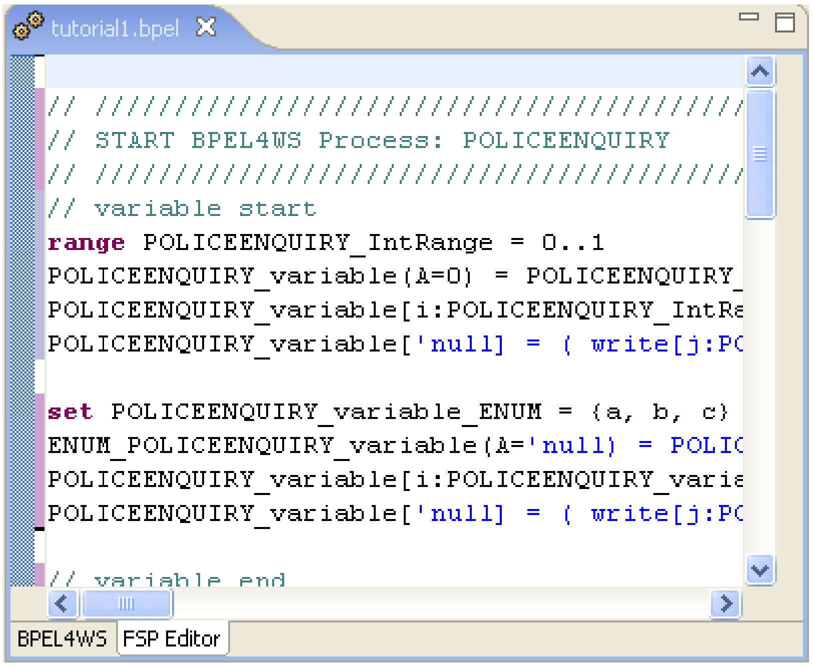


c. This option will open the file as a LTSA BPEL Editor file and display the contents in a new Editor View, as illustrated below.



```
<process name="PoliceEnquiry"
         targetNamespace="urn:pito:PoliceEnqu
         xmlns:tns="urn:pito:PoliceEnquiry"
         xmlns="http://schemas.xmlsoap.org/ws,

   <variables>
      <variable name="request" messageType="t
   </variables>

   <partners>
      <partner name="OfficerDevice" serviceLi
      <partner name="VehicleEnquiry" serviceL
      <partner name="NominalEnquiry" serviceL
      <partner name="InsuranceEnquiry" service
```

1. Note the tabs along the bottom of the BPEL4WS Editor View. The **BPEL4WS** tab is the main editor for the BPEL4WS source, whilst the **FSP Editor tab** can be selected to show a FSP representation of the BPEL4WS.
2. Click on the **FSP Editor tab** to list the FSP for the BPEL4WS currently being edited.

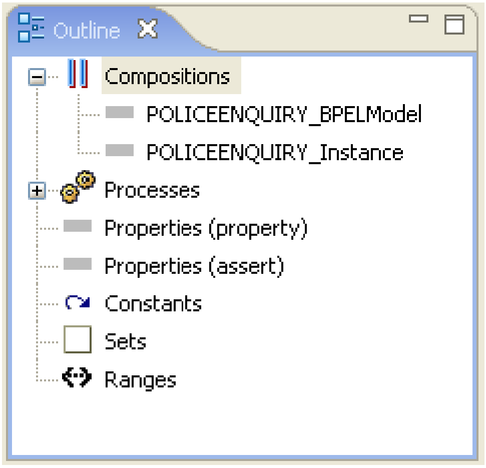


```
// ////////////////////////////////////////
// START BPEL4WS Process: POLICEENQUIRY
// ////////////////////////////////////////
// variable start
range POLICEENQUIRY_IntRange = 0..1
POLICEENQUIRY_variable(A=0) = POLICEENQUIRY_
POLICEENQUIRY_variable[i:POLICEENQUIRY_IntRa
POLICEENQUIRY_variable['null] = ( write[j:PO

set POLICEENQUIRY_variable_ENUM = {a, b, c}
ENUM_POLICEENQUIRY_variable(A='null) = POLIC
POLICEENQUIRY_variable[i:POLICEENQUIRY_varia
POLICEENQUIRY_variable['null] = ( write[j:PO

// variable end
```

3. Note that whenever the BPEL4WS source changes, the switch to FSP will update the FSP source code representing the new BPEL4WS source. Try to change something in the BPEL4WS Editor and switch to see any changes in the FSP.

Compile the LTS

1. Note that the Outline View now contains some elements of the BPEL4WS process. The core compositions are a _**BPELModel** and _**Instance**. The _**BPELModel** composition represents all possible trace paths through the BPEL4WS process, whilst _**Instance** represents only one (default) choice.



2. The user can analyse these processes from the **WS-BPEL** source by using the same approach described in the LTSA Eclipse help

END OF TUTORIAL

3.1.3 WS-CDL and LTSA-WSCDL Editor

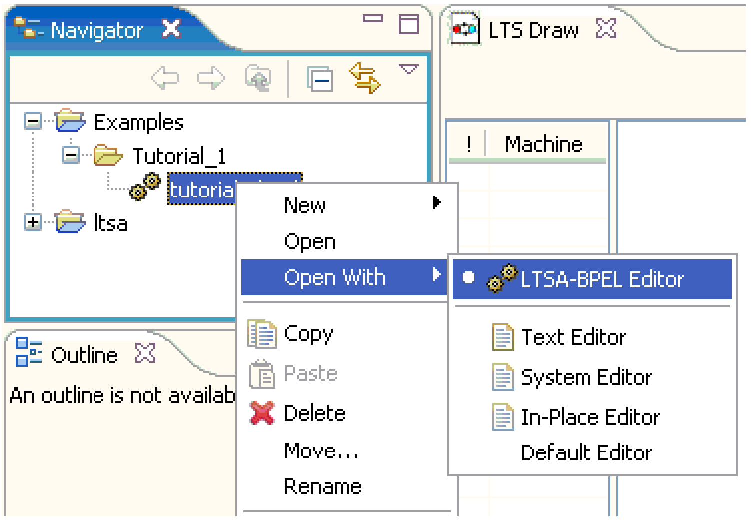
In this tutorial the user will be able to import a WS-CDL description and generate a Finite State Process composition from the description.

1. Import WS-CDL files in to a project

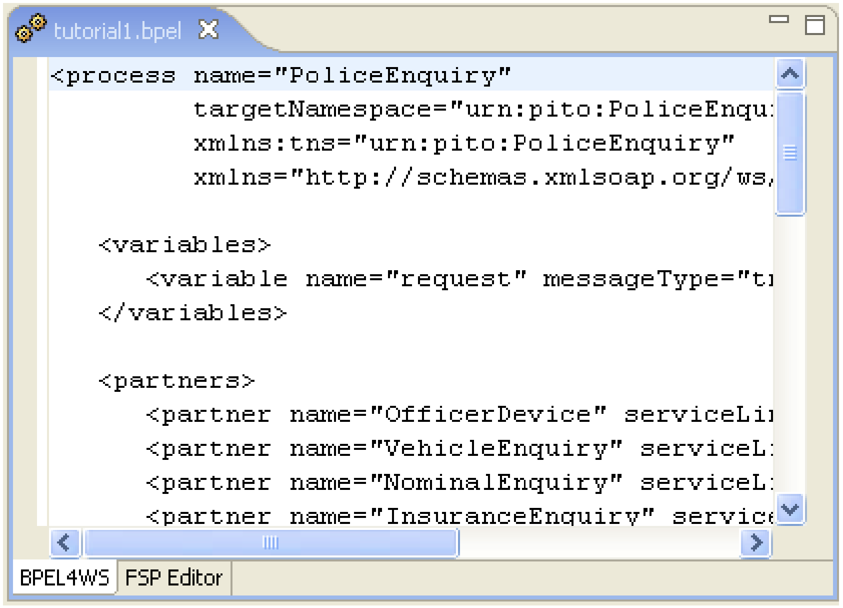
a. Locate your WS-CDL source files and import them in to the project created previously.
b. In this example we use a WS-CDL description called "**tutorial1.bpel**".
c. You can substitute the name of your WS-CDL files for this one at each step.

2. Open the WS-CDL file in WS-Engineer

1. Locate the **WS-CDL** file (with extension .bpel) in the Navigator view.
2. Select the file, and right-click the mouse, select **Open With** → **LTSA-BPEL Editor,** as illustrated below.

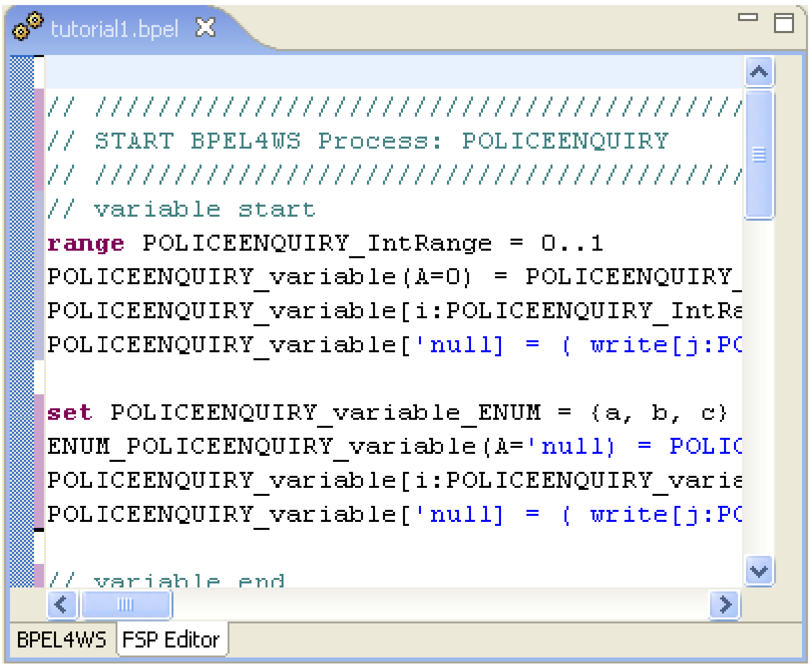


3. This option will open the file as a LTSA BPEL Editor file and display the contents in a new Editor View, as illustrated below.



```
<process name="PoliceEnquiry"
         targetNamespace="urn:pito:PoliceEnqu
         xmlns:tns="urn:pito:PoliceEnquiry"
         xmlns="http://schemas.xmlsoap.org/ws/

   <variables>
      <variable name="request" messageType="t
   </variables>

   <partners>
      <partner name="OfficerDevice" serviceLi
      <partner name="VehicleEnquiry" serviceL
      <partner name="NominalEnquiry" serviceL
      <partner name="InsuranceEnquiry" service
```

4. Note the tabs along the bottom of the BPEL4WS Editor View. The **BPEL4WS** tab is the main editor for the BPEL4WS source, whilst the **FSP Editor tab** can be selected to show a FSP representation of the BPEL4WS.
5. Click on the **FSP Editor tab** to list the FSP for the BPEL4WS currently being edited.



```

// ////////////////////////////////////////////
// START BPEL4WS Process: POLICEENQUIRY
// ////////////////////////////////////////////
// variable start
range POLICEENQUIRY_IntRange = 0..1
POLICEENQUIRY_variable(A=0) = POLICEENQUIRY_
POLICEENQUIRY_variable[i:POLICEENQUIRY_IntRa
POLICEENQUIRY_variable['null] = ( write[j:PO

set POLICEENQUIRY_variable_ENUM = {a, b, c}
ENUM_POLICEENQUIRY_variable(A='null) = POLIC
POLICEENQUIRY_variable[i:POLICEENQUIRY_varia
POLICEENQUIRY_variable['null] = ( write[j:PO

// variable end
```

6. Note that whenever the BPEL4WS source changes, the switch to FSP will update the FSP source code representing the new BPEL4WS source. Try to change something in the BPEL4WS Editor and switch to see any changes in the FSP.

END OF TUTORIAL

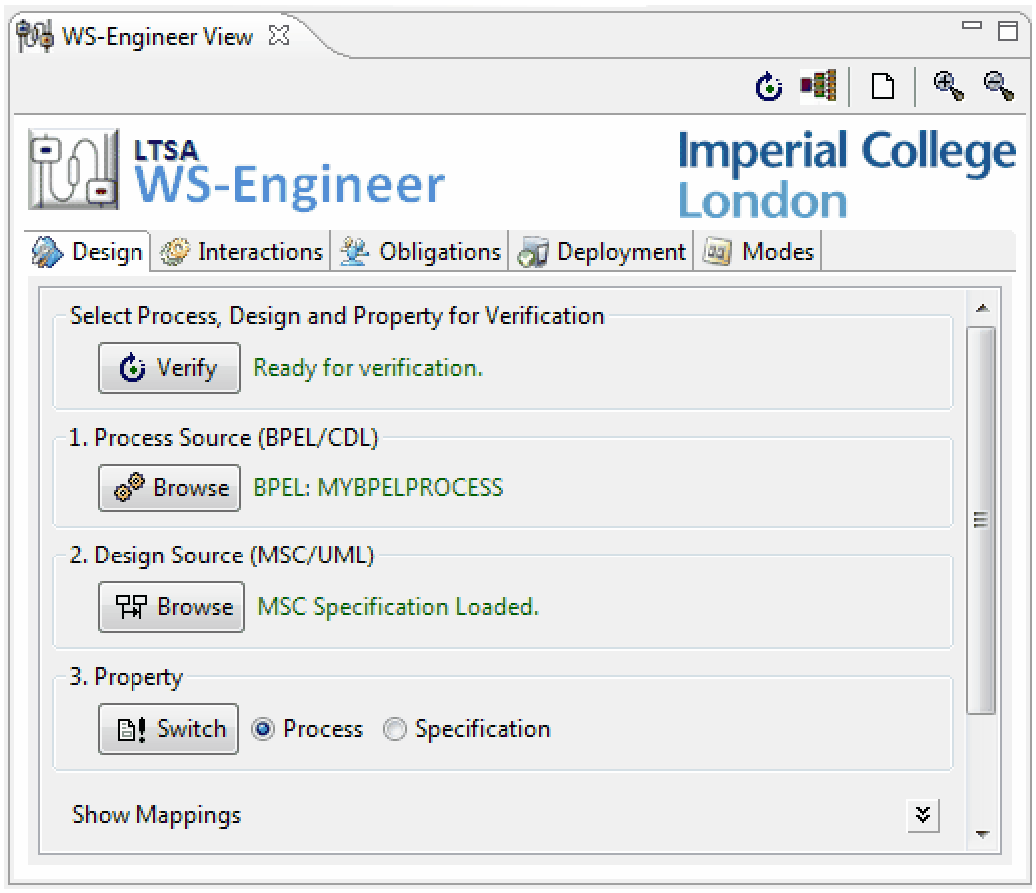
3.2 Design Analysis

This section describes various common activities when using the Design Check features in the WS-Engineer plug-in.

3.2.1 Overview

The design view of WS-Engineer provides verification of WS-BPEL orchestration process against that of design specifications in the form of UML style sequence charts. We model the interaction sequences built to support multiple-partner conversations across enterprise domains and with a view of wider goals. We limit the scope of choreography analysis to that of interactions between compositions.

The design view can be accessed by selecting the Design tab on the WS-Engineer view tab list.



Select Process, Design and Property for Verification

Ready for verification.

1. Process Source (BPEL/CDL)

BPEL: MYBPELPROCESS

2. Design Source (MSC/UML)

MSC Specification Loaded.

3. Property

[✓ Process] Specification

The essence of this verification is to prove that a property exists in the composition modelling of combined implementation and design models. Although the task of comparing models is easier in simple processes, more complex processes require an in-depth and time-consuming comparison. Model checking can then be performed to formally test that a WS-BPEL implementation provides the necessary activities to meet the MSC specification, through a model trace. Additionally, the aim is to provide as much a mechanical verification as possible, so that observation is not required by human eye in larger more complex processes.

You can read more about this type of verification in Chapter 6 of "**A Rigorous Approach to Engineering Web Service Compositions**" by Howard Foster Imperial College London.

3.2.2 Preparation

For this tutorial you need to have met several pre-requisites:

1. Created a Design Specification for a Web Service Composition as MSCs using the LTSC-MSC Editor.
2. Created a WS-BPEL process (or obtained an existing WS-BPEL process representing the design specification created in 1.).

These inputs are required to perform the verification step.

There are Sample Files which can be used as source files described in this tutorial.

3.2.3 Tutorial: Verifying Design and WS-BPEL

In this tutorial the user will be able to analyse a WS-BPEL process for fulfillment of a Design Specification.

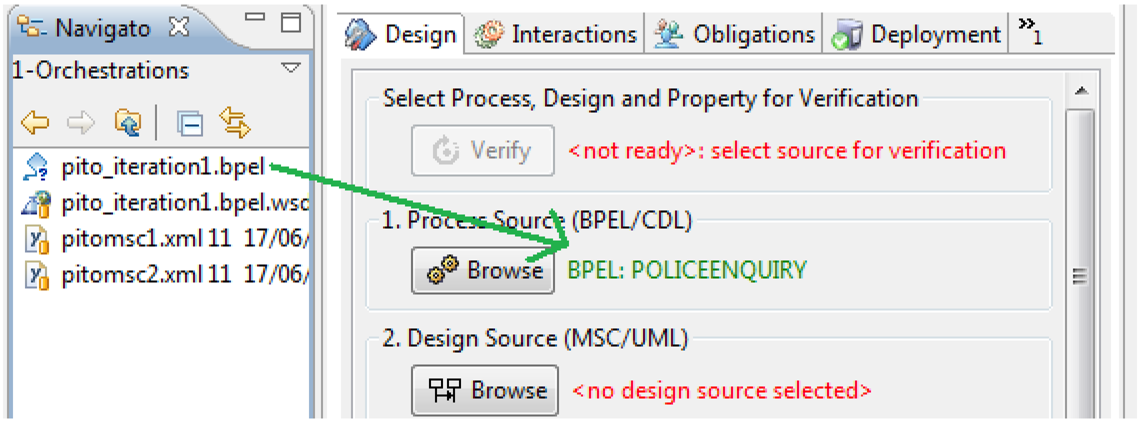
Note: Ensure you have carried out the pre-requisites as described in the Preparation section.

Steps

1. **Select the** Design **tab on the WS-Engineer View**

2. **Add a BPEL process**

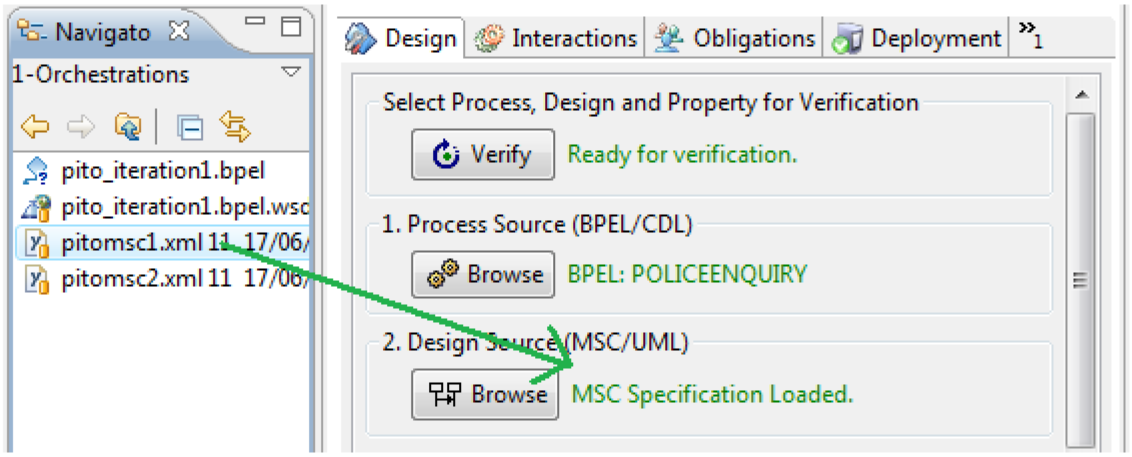
Click on the Browse button to select the WS-BPEL process, alternatively you can "drag and drop" the .bpel file on the Design View as shown below.



If the WS-BPEL file is valid, then the name of the BPEL process is shown under the 1. Process Source section.

3. Select the Design Source

Click on the button to select the Design (MSC) specification, alternatively you can "drag and drop" the .xml file on the Design View as shown below.



If the Design file is valid, then "**MSC Specification Loaded**" is shown under the 2. Design Source section.

4. Check the property of interest for analysis

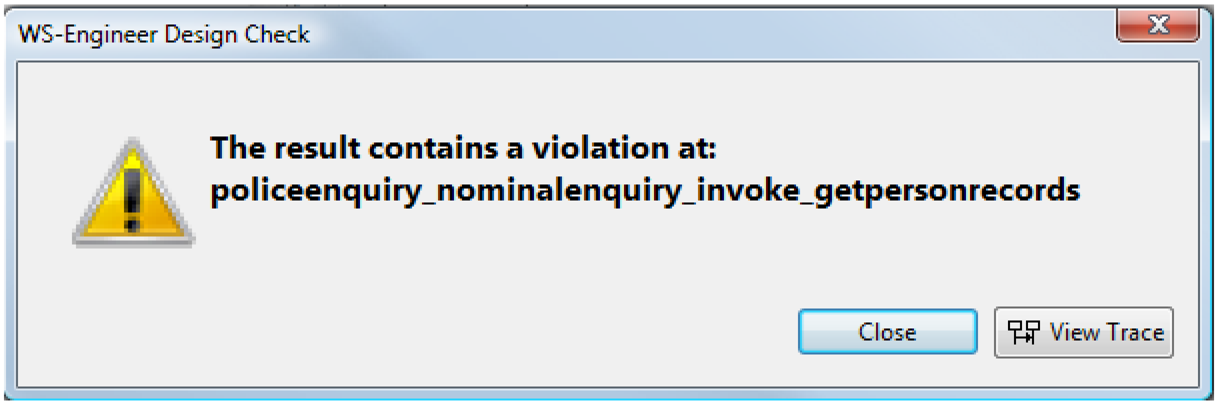
Click the button select the source property for verification.

- Property as **Process** uses the WS-BPEL implementation as a property against the design specification (e.g. does the implementation fulfill the behaviour specified in the design specification).
- Property as **Specification** uses the MSC Design Specification as a property against the WS-BPEL implementation (e.g. does the specification exhibit any additional behaviour not specified in the implementation).

5. Run the Verify action

When both implementation and specification files are correctly loaded, the Verify button will be enabled. Click the button to start the verification. If the button is not enabled after adding both valid process and design files, then it is possible that the behaviour mappings between these are not complete and must be manually assigned.

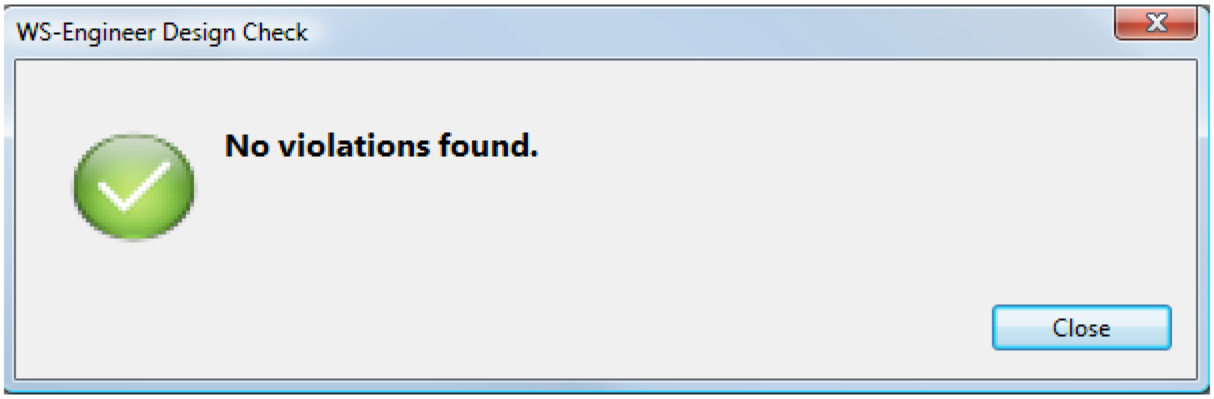
If there is behaviour found that is not equivalent (a property violation) then the point at which the violation occurs is shown as a warning.


WS-Engineer Design Check

The result contains a violation at:
policeenquiry_nominalenquiry_invoke_getpersonrecords

You can click the View Trace button to show the trace to violation as an MSC.

If the implementation and specification are equivalent (according to the property selected) then **No Violations found** will be shown.


WS-Engineer Design Check

No violations found.

END OF TUTORIAL

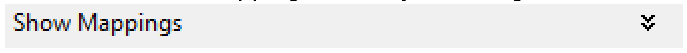
3.2.4 Options

3.2.4.1 Behaviour Mappings

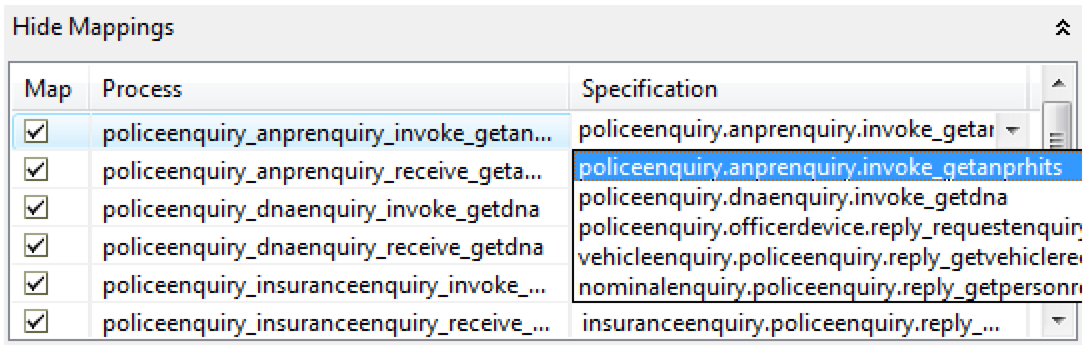
Show or Modify the mappings between Process and Design Specification

WS-Engineer automatically attempts to match behaviour labels from the WS-BPEL implementation and messages in the Design Specification MSC. However, messages in the Design Specification may not exactly match depending on the choice of labels by the designer.

You can see the mappings used by selecting



bar at the bottom of the view.

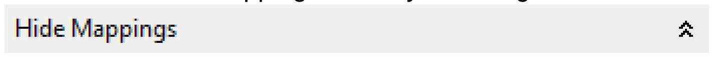


Hide Mappings

Map	Process	Specification
✓	policeenquiry_anprenquiry_invoke_getan...	policeenquiry.anprenquiry.invoke_getar
✓	policeenquiry_anprenquiry_receive_geta...	policeenquiry.anprenquiry.invoke_getanprhits
✓	policeenquiry_dnaenquiry_invoke_getdna	policeenquiry.dnaenquiry.invoke_getdna
✓	policeenquiry_dnaenquiry_receive_getdna	policeenquiry.officerdevice.reply_requestenquir
✓	policeenquiry_insuranceenquiry_invoke_...	vehicleenquiry.policeenquiry.reply_getvehiclere
✓	policeenquiry_insuranceenquiry_receive_...	nominalenquiry.policeenquiry.reply_getpersonr
		insuranceenquiry.policeenquiry.reply_...

The specification mappings can the be selected representing particular actions in the WS-BPEL process.

You can hide the mappings used by selecting



bar above the mappings list.

3.2.5 Further Reading

Read the following papers on the techniques used for design analysis of web service compositions.

- H.Foster, S.Uchitel, J.Magee, J.Kramer, Model-based Verification of Web Service Compositions, IEEE Automated Software Engineering (ASE) 2003, Montreal, Canada, 2003.
- H.Foster A Rigorous Approach to Engineering Web Service Compositions, PhD Thesis, Imperial College London, January, 2006.

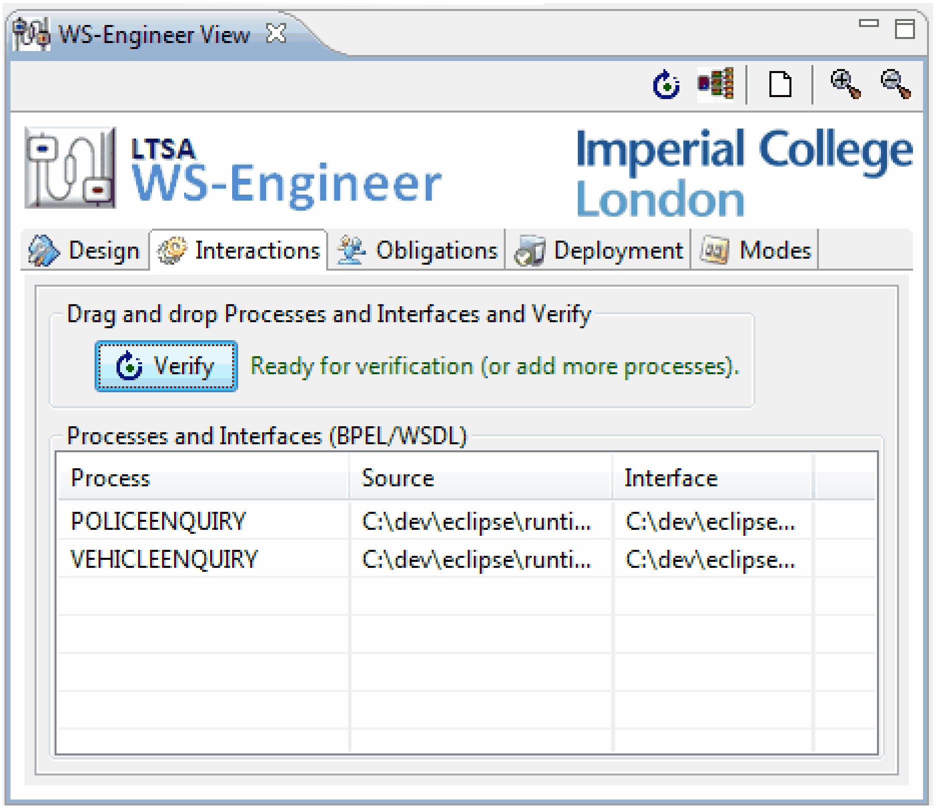
3.3 Interactions Analysis

This section describes various common activities when using the Interaction Checking features in the WS-Engineer plug-in.

3.3.1 Overview

The interactions view of WS-Engineer provides verification of collaborating WS-BPEL orchestration processes. We model the interaction sequences built to support multiple-partner conversations across enterprise domains and with a view of wider goals. We limit the scope of choreography analysis to that of interactions between compositions.

The interactions view can be accessed by selecting the Interactions tab on the WS-Engineer view tab list.



Drag and drop Processes and Interfaces and Verify

Ready for verification (or add more processes).

Processes and Interfaces (BPEL/WSDL)

Process	Source	Interface
POLICEENQUIRY	C:\dev\eclipse\runti...	C:\dev\eclipse...
VEHICLEENQUIRY	C:\dev\eclipse\runti...	C:\dev\eclipse...

The verification performs a service "port connector" based mapping. The verification ensures that the interacting partners of the service composition fulfill required interaction behaviour, in the correct sequence and over the correct channels.

You can read more about this type of verification in Chapter 6 of "**A Rigorous Approach to Engineering Web Service Compositions**" by Howard Foster Imperial College London.

3.3.2 Preparation

For this tutorial you need to have met several pre-requisites:

1. Created at least two interacting WS-BPEL processes (or obtained existing WS-BPEL processes).
2. Created at least two WSDL specifications for the interacting processes in 1.

These inputs are required to perform the verification step.

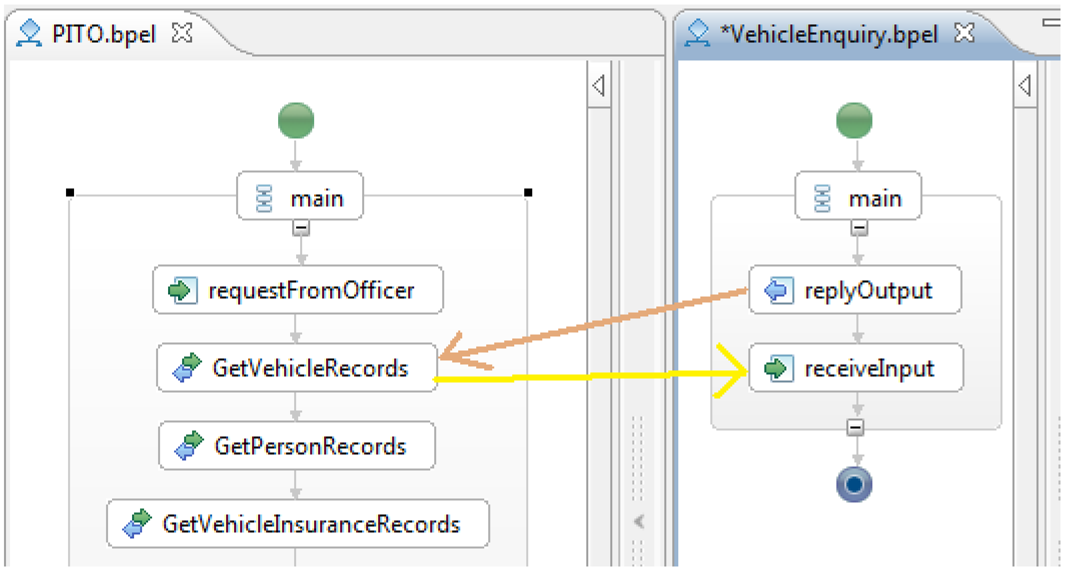
There are Sample Files which can be used as source files described in this tutorial.

3.3.3 Tutorial: Interaction Analysis

In this tutorial the user will be able to analyse interacting WS-BPEL processes for completeness and correctness of behaviour specified.

Note: Ensure you have carried out the pre-requisites as described in the Preparation section.

The verification performs a service "port connector" based mapping. The verification ensures that the interacting partners of the service composition fulfill required interaction behaviour, in the correct sequence and over the correct channels.In a simple example, the two WS-BPEL processes below are executed out of sequence, as the VehicleEnquiry partner firstly executes a reply and then a receive (receive should be before the reply).



In this tutorial we will analyse these processes, observe the violation and then correct the partner process to observe correctness.

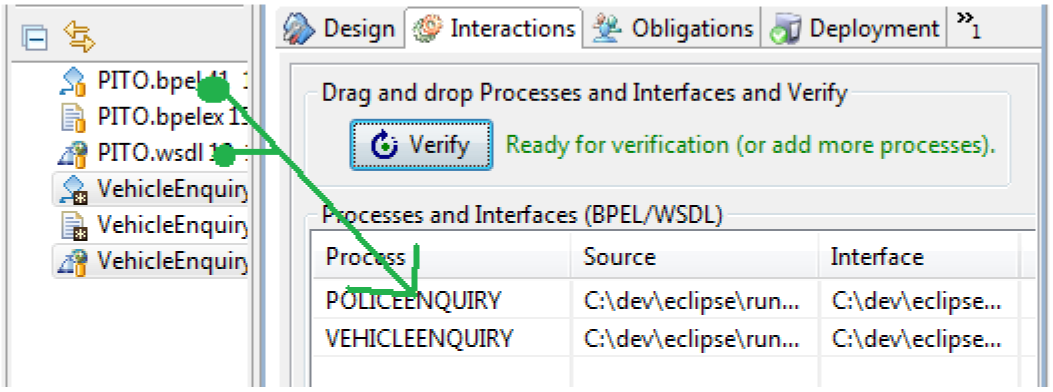
Steps

1. **Select the** Interactions **Tab on the WS-Engineer View**

2. **Select the WS-BPEL and WSDL process pairs to analyse**

Drag and drop selected WS-BPEL and WSDL files (together) in the Processes and Interfaces list box of the interactions view. Repeat this for as many interacting processes as you wish to analyse.

Note: Each WS-BPEL process is paired with one or more WSDL files. In this sample, we associate one WS-BPEL with one WSDL file.



Drag and drop Processes and Interfaces and Verify

Verify   Ready for verification (or add more processes).

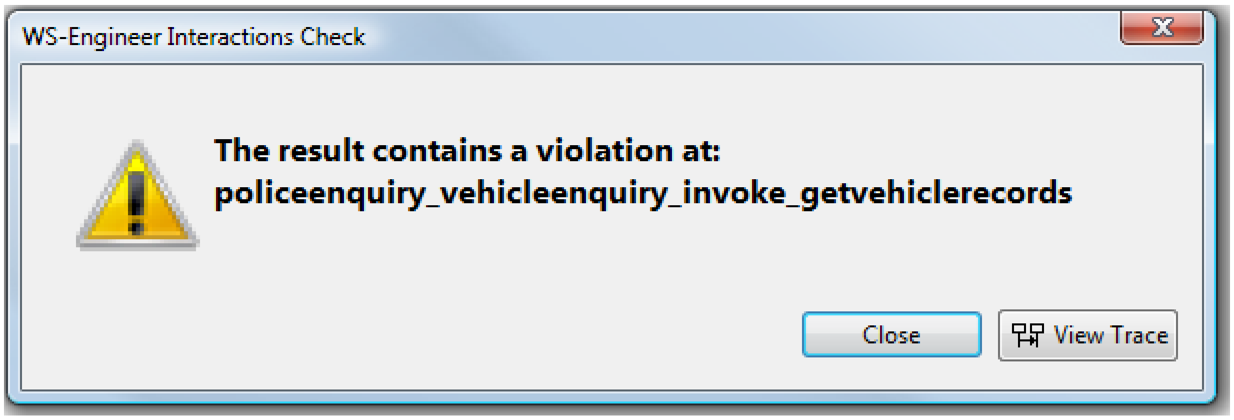
Processes and Interfaces (BPEL/WSDL)

Process	Source	Interface
POLICEENQUIRY	C:\dev\eclipse\run...	C:\dev\eclipse...
VEHICLEENQUIRY	C:\dev\eclipse\run...	C:\dev\eclipse...

3. Run the Verify action

When two or more processes and WSDL files have been added, the Verify button will be enabled. Click the button to start the interaction verification.
Note: If the button is not enabled after adding both valid process and WSDL files, then it is possible that the either of the file types is not valid. Check each file for the correct syntax and form.

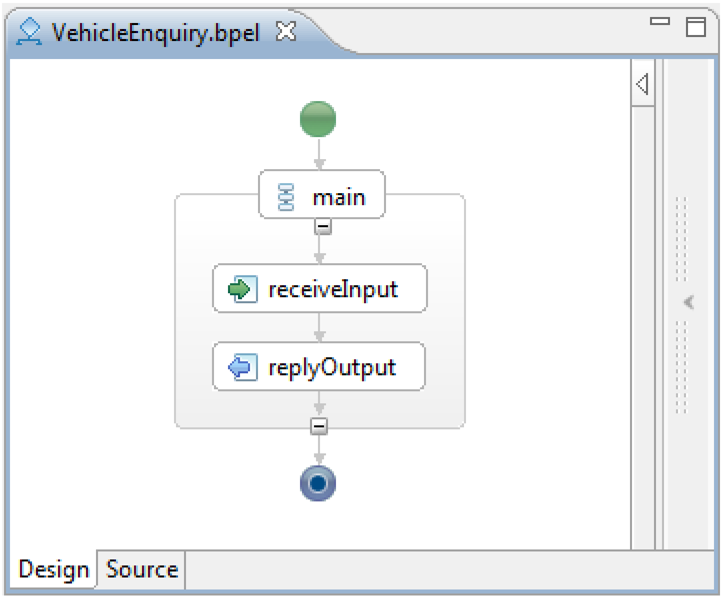
In the first case (of the out of sequence partner described previously) the following violation is observed.


WS-Engineer Interactions Check

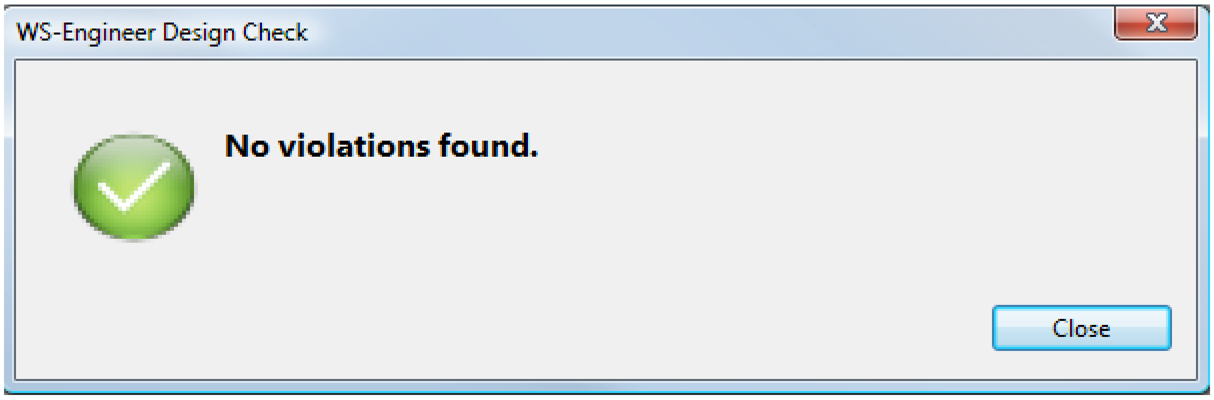
The result contains a violation at: policeenquiry_vehicleenquiry_invoke_getvehiclerecords

You can click the View Trace button to show the trace to violation as an MSC.

Now, change the VehicleEnquiry process to reflect the receive BEFORE the reply (as shown below).



Repeating the verification step shows that **No Violations found**.


WS-Engineer Design Check

No violations found.

END OF TUTORIAL

3.3.4 Further Reading

Read the following papers on the techniques used for interaction analysis of web service compositions.

- H.Foster, S.Uchitel, J.Magee, J.Kramer, Compatibility Verification for Web Service Choreography, IEEE International Conference on Web Services (ICWS) 2004, San Diego, CA, July 2004.
- H.Foster A Rigorous Approach to Engineering Web Service Compositions, PhD Thesis, Imperial College London, January, 2006.

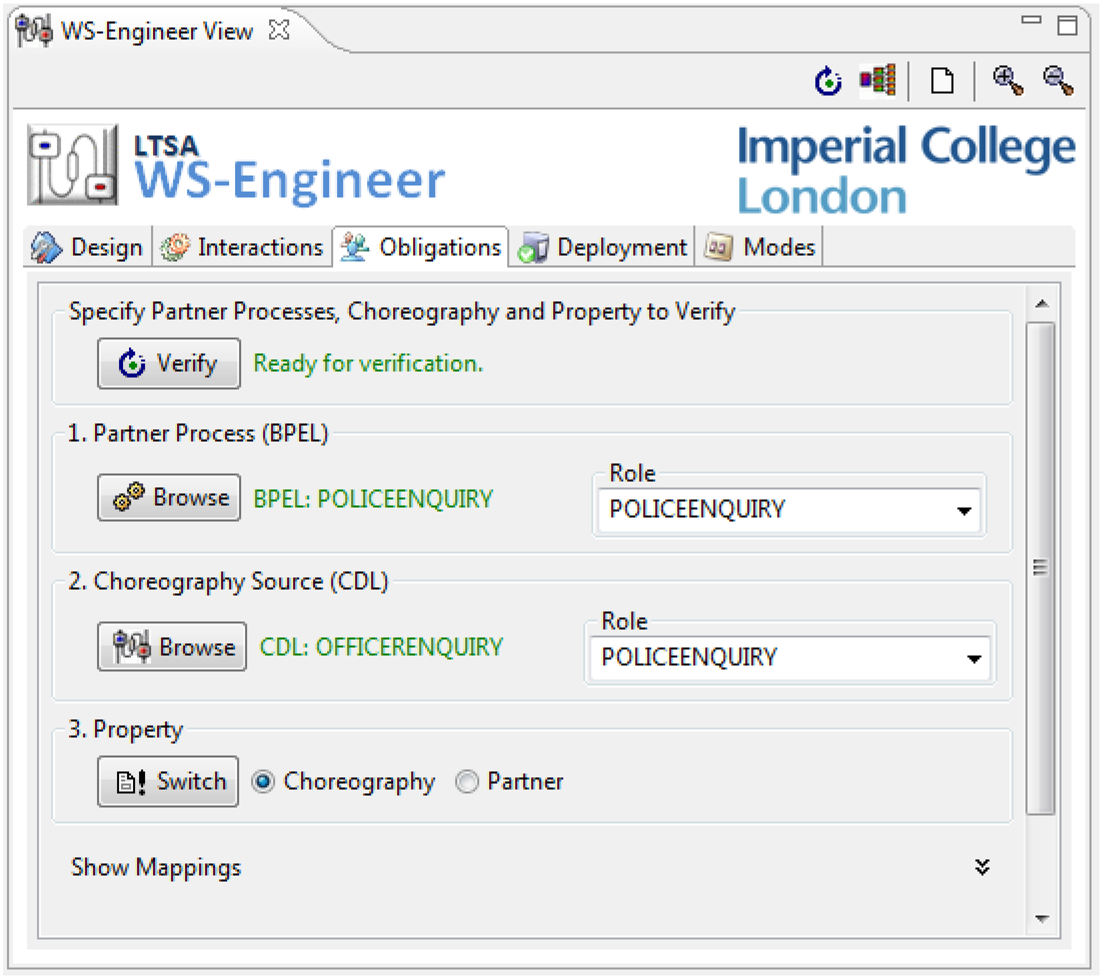
3.4 Obligations Analysis

This section describes various common activities when using the Obligations Checking features in the WS-Engineer plug-in.

3.4.1 Overview

The obligations view of WS-Engineer provides verification of orchestration partners (roles in the form of WS-BPEL) of a choreography policy (in the form of WS-CDL). The verification provides a mechanism to check that each partner in the choreography fulfills their obligations as part of a role of the choreography.

The obligations view can be accessed by selecting the Obligations tab on the WS-Engineer view tab list.



Specify Partner Processes, Choreography and Property to Verify

Ready for verification.

1. Partner Process (BPEL)

BPEL: POLICEENQUIRY — Role: POLICEENQUIRY

2. Choreography Source (CDL)

CDL: OFFICERENQUIRY — Role: POLICEENQUIRY

3. Property

[✓ Choreography] Partner

s

3.4.2 Preparation

For this tutorial you need to have met several pre-requisites:

1. Created a service choreography policy in the form of WS-CDL.
2. Created a WS-BPEL process (or obtained an existing WS-BPEL process representing the role of a

partner in the choreography created in 1.).

These inputs are required to perform the verification step.

There are Sample Files which can be used as source files described in this tutorial.

3.4.3 Tutorial: Obligations Analysis

In this tutorial the user will be able to analyse partner interactions (in the form of WS-BPEL processes) against their obligations as part of a service choreography (in the form of WS-CDL).

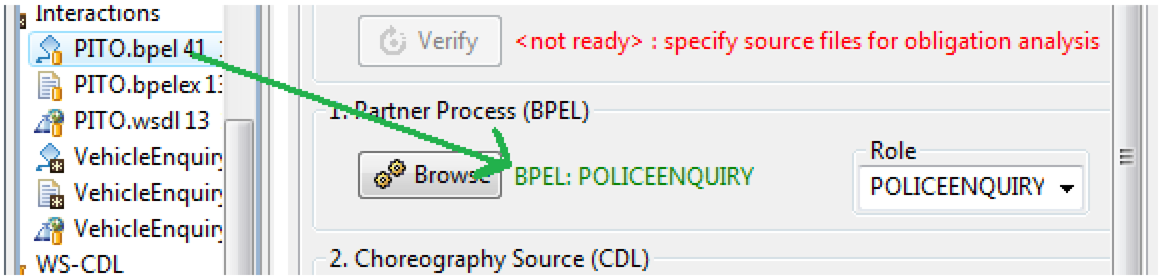
Note: Ensure you have carried out the pre-requisites as described in the Preparation section.

Steps

1. **Select the Obligations Tab on the WS-Engineer View**

2. **Add the Partner Process (WS-BPEL) to the Obligations page**

Click on the Browse button to select the WS-BPEL process, alternatively you can "drag and drop" the .bpel file on the Obligations page as shown below.
If the file is valid the BPEL process name will appear in 1. Partner Process, and a default role will be added to the Process Role List.

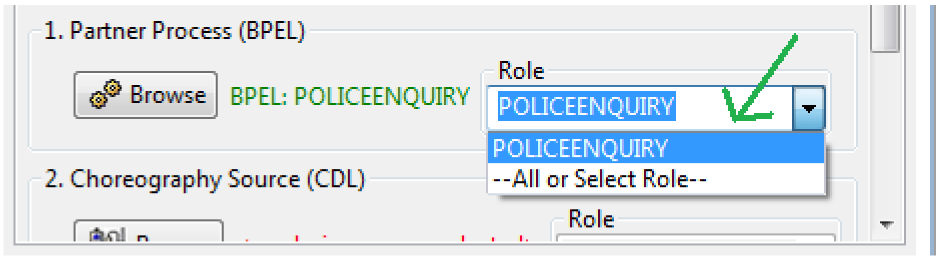


<not ready> : specify source files for obligation analysis

BPEL: POLICEENQUIRY

3. **Select the Partner Process Role**

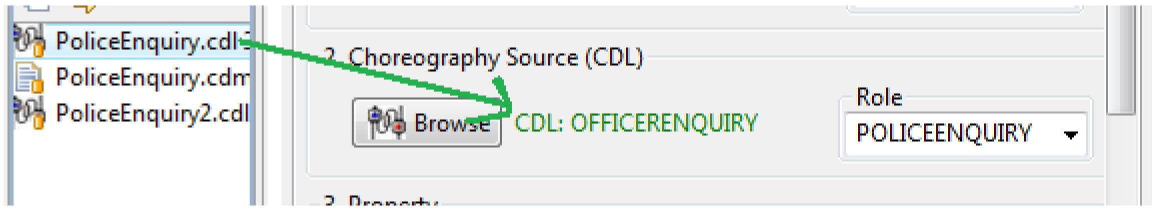
Expanding the Role list aside the Partner Process name will list all available roles of this process.
Note: the BPEL process can act as several partner roles. The **default** is the partner process name.



BPEL: POLICEENQUIRY

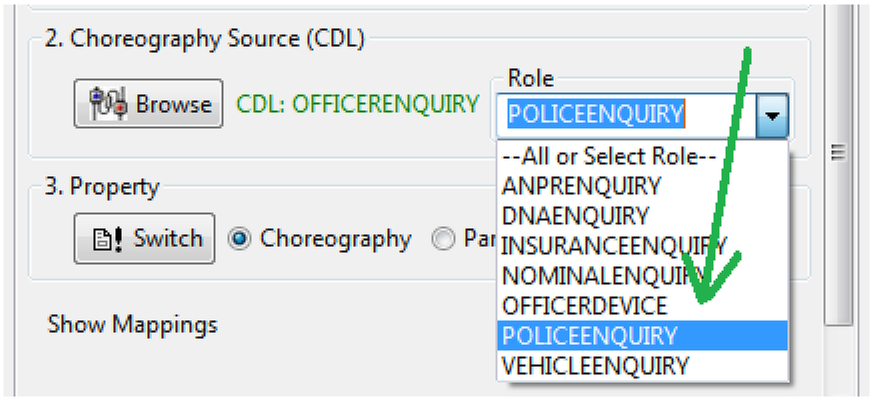
4. Add the Choreography Policy specification (WS-CDL) to the Obligations Page

Click on the button to select a WS-CDL choreography specification, alternatively you can "drag and drop" the .cdl file on the Obligations page as shown below.
If the file is valid the CDL choreography name will appear in 2. Choreography source, and a default role will be selected in the Choreography Role List.

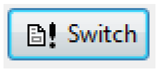


5. Select the Choreography Partner Role

Expanding the Role list aside the Choreography Source name will list all available roles defined in this choreography.
Note: The default role is chosen by the selection of the **Partner Process Role** in step 2.



6. Check the property of interest for analysis

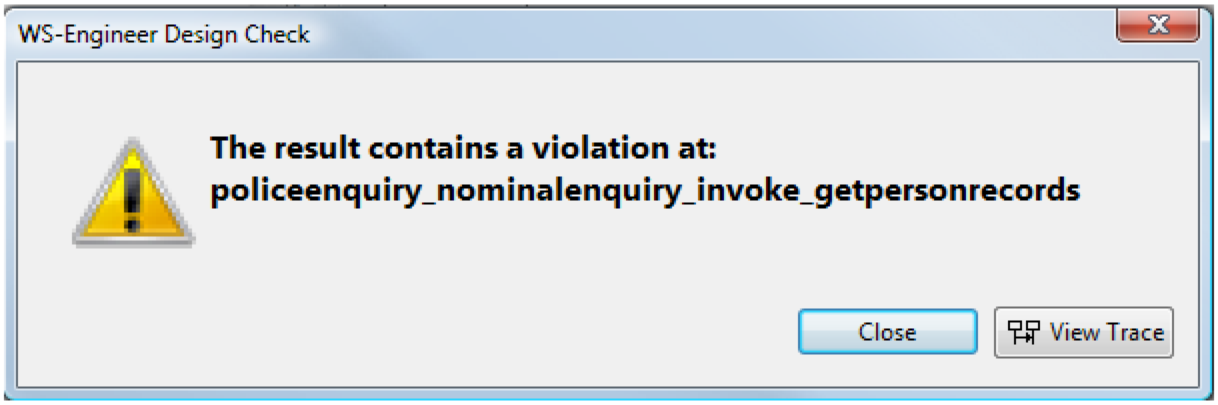
Click the  button select the source property for verification.

- Property as **Choreography** uses the choreography specification as a property against the partner process role (e.g. does the implementation fulfill the behaviour specified in the choreography specification).
- Property as **Partner** uses the partner process role as a property against the choreography specification implementation (e.g. does the specification exhibit any additional behaviour for the role that is not specified in the implementation).

7. Run the Verify action

When both implementation and choreography files are correctly loaded, the Verify button will be enabled. Click the button to start the verification. If the button is not enabled after adding both valid process and choreography files, then it is possible that the behaviour mappings between these are not complete and must be manually assigned.

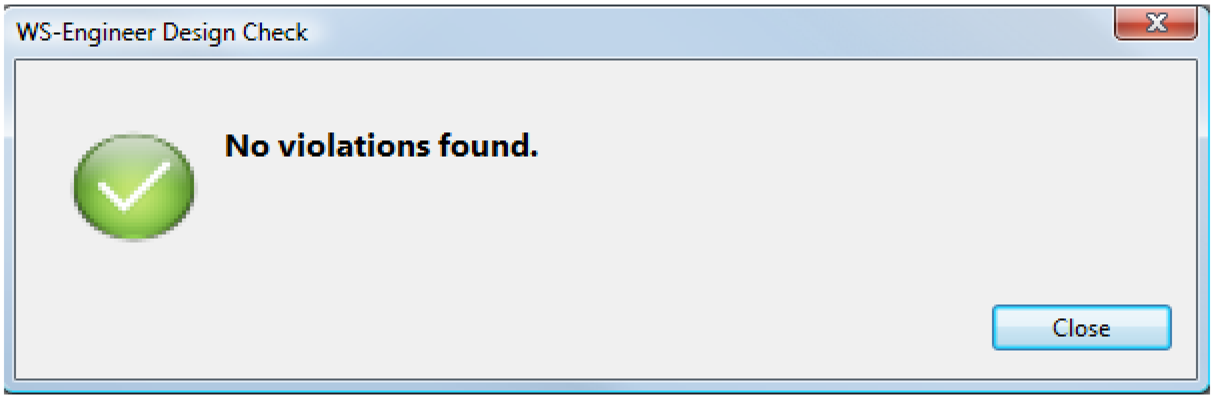
If there is behaviour found that is not equivalent (a property violation) then the point at which the violation occurs is shown as a warning.


WS-Engineer Design Check

The result contains a violation at: policeenquiry_nominalenquiry_invoke_getpersonrecords

You can click the View Trace button to show the trace to violation as an MSC.

If the implementation and specification are equivalent (according to the property selected) then **No Violations found** will be shown.


WS-Engineer Design Check

No violations found.

END OF TUTORIAL

3.4.4 Options

This section contains additional tasks in Obligations Analysis.

3.4.5 Further Reading

Read the following papers on the techniques used for obligations analysis of web service compositions.

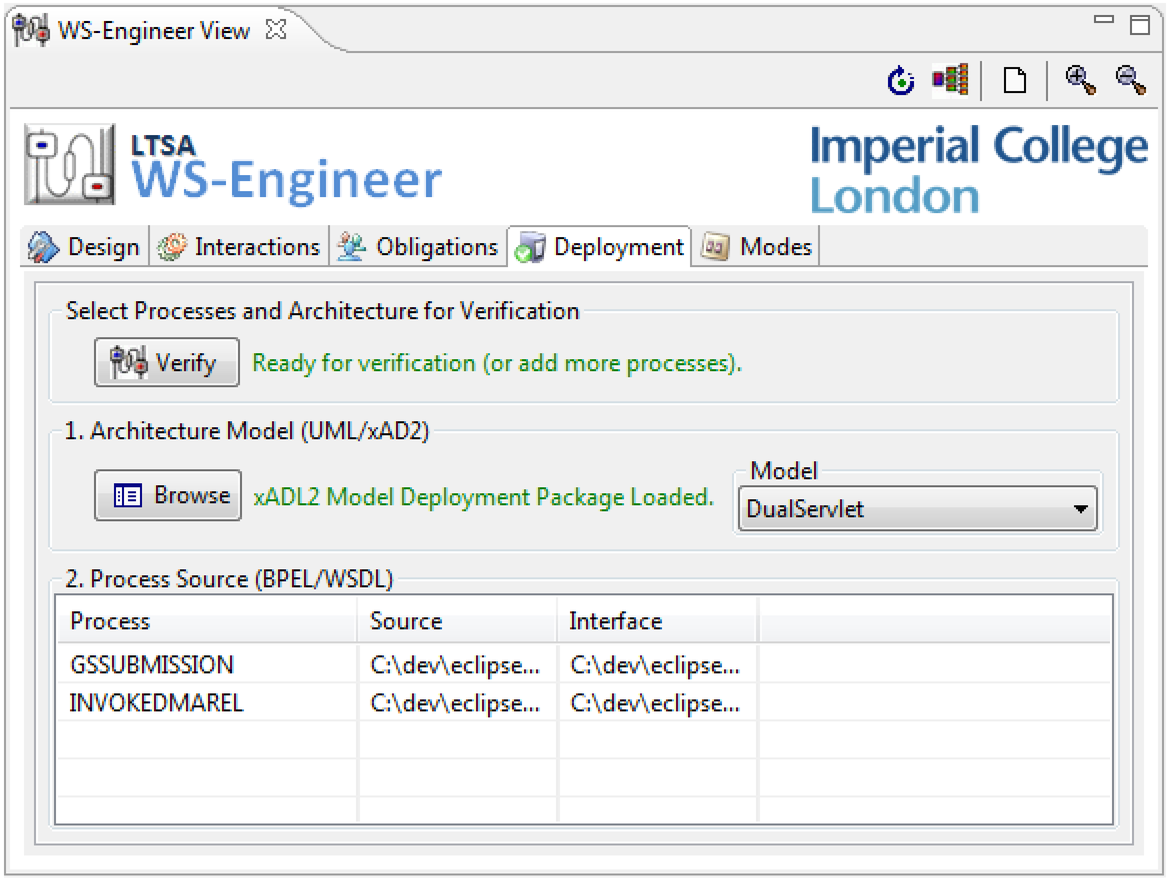
- H.Foster, S.Uchitel, J.Magee, J.Kramer, Model-Based Analysis of Obligations in Web Service Choreography, IEEE International Conference on Internet&Web Applications and Services (ICIW) 2006, Guadeloupe, French Caribbean
- H.Foster A Rigorous Approach to Engineering Web Service Compositions , PhD Thesis, Imperial College London, January, 2006.

3.5 Deployment Analysis

This section describes various common activities when using the Deployment Checking features in the WS-Engineer plug-in.

3.5.1 Overview

The deployment view of WS-Engineer provides verification of orchestration partners and properties of service deployment architectures (such as resource allocations). The verification provides a mechanism to check that the choice of composition behaviour design, architecture configurations and resource allocations do not cause undesirable deadlock situations.



Select Processes and Architecture for Verification

Verify — Ready for verification (or add more processes).

1. Architecture Model (UML/xAD2)

Browse — xADL2 Model Deployment Package Loaded.

Model: DualServlet

2. Process Source (BPEL/WSDL)

Process	Source	Interface
GSSUBMISSION	C:\dev\eclipse...	C:\dev\eclipse...
INVOKEDMAREL	C:\dev\eclipse...	C:\dev\eclipse...

You can read more about this type of verification in the following publication: H.Foster, W.Emmerich, J.Kramer, J.Magee, D.Rosenblum and S.Uchitel, **Model Checking Service Compositions under Resource Constraints**, in Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), Dubrovnik, Croatia. Sept 2007.

3.5.2 Preparation

For this tutorial you need to have met several pre-requisites:

1. Created one or more XADL2.0 Deployment Models or a UML2 Deployment Diagram specifications.
2. Created one or more WS-BPEL processes and their associated WSDL interface specifications.

The sections of Preparation illustrate how to create these pre-requisites.

There are Sample Files which can be used as source files described in this tutorial.

3.5.2.1 Create a xADL2 Service Deployment Architecture

This is an example of how the xADL2 Service Deployment configuration was created for the Molpak orchestration, for both SingleServet and DualServlet configurations.

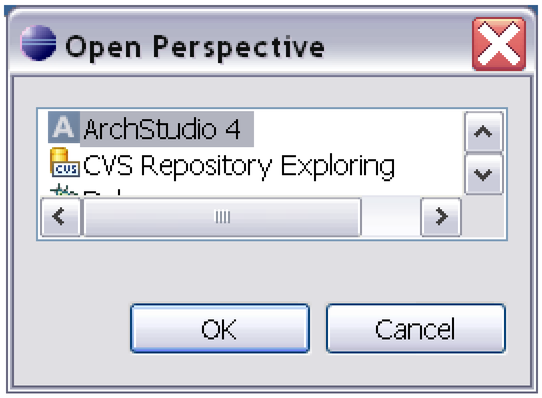
Prerequisites

Plug-in Check List (see install instructions)

- ArchStudio 4 (to create new xADL2 architecture configurations)

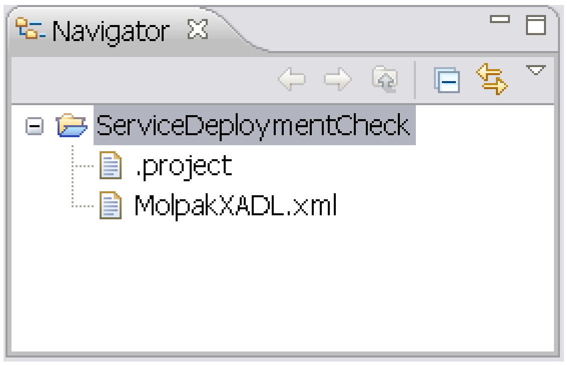
1. Switch to the ArchStudio 4 Perspective

a. Locate the **Eclipse menu bar**.
b. Select **Window** → **Open Perspective** → **Other**.
c. Select **ArchStudio 4** and click **OK**.



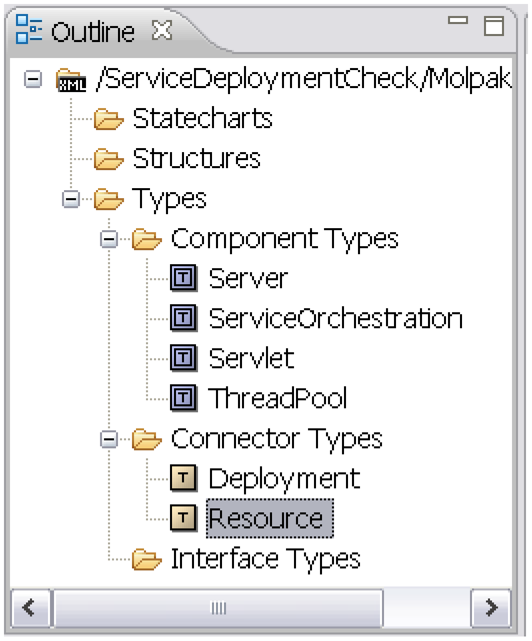
1. Create a new xADL Architecture Description

a. Select the new project **ServiceDeploymentCheck**.
b. Select **File** → **New** → **Other** → **ArchStudio4** → **ArchStudio Architecture Description** and click **next.**
c. Name the file **MolpakXADL** and click **Finish.**



2. Create the Service Deployment Stereotypes

a. Select the new file **MolpakXADL.xml** in the project **ServiceDeploymentCheck**.
b. Right-click with mouse, and select **Open-With → Archipelago.**
c. Locate the Outline View, and select the parent item "**/ServiceDeploymentCheck/Molpak.XADL**", right-click and select **Create Type Set**.
d. Open the new **Types** branch and select **Component Types**.
e. Right-click and select **New Component Type**. Select the **[New Component Type]** added under Component Types. Right-click and select **Edit Description**.
f. Type the name as **Server.**
g. Repeat e. and g. for the following Component Types: **ServiceOrchestration**, **Servlet** and **ThreadPool**.
h. Create new **Connector Types** for: **Deployment** and **Resource**.

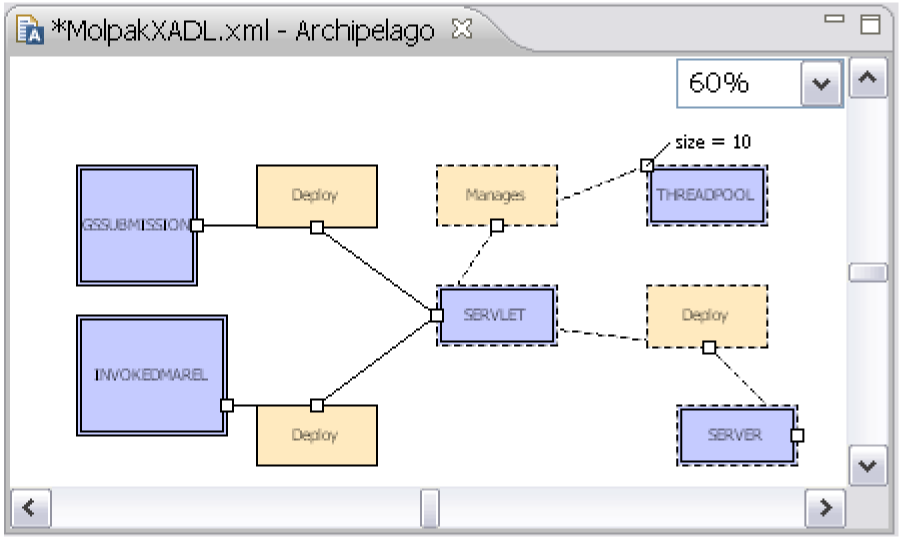


3. **Create a Single Servlet Deployment Architecture**

a. Select the **Structures** branch, right-click and select **New Structure**.
b. Select the **[New Structure]** item, right-click and edit the description to be **SingleServlet.**
c. Double-click the new **SingleServlet** structure to open the **Archipelago Structure Editor**.
d. Locate the **Component Types** branch in the **Outline View** and **drag-and-drop** a number of each type to the **Structure Editor**. For the molpak example, we need: **2x ServiceOrchestration**, **1x ThreadPool**, **1x Servlet**, **1x Server**, **3x Deployment** and **1x Resource**.
e. For **ServiceOrchestration components**, right-click and edit the description to the name of each BPEL In the case of the molpak example, this is one for **GSSubmission** and one for **InvokedMarel**.
f. On each component and connector, create a **New Interface** (right-click each instance).
g. For **ServiceOrchestrations**, create a **new link** between the ServiceOrchestration interfaces and a **Deployment connector** interface.
h. For the ServiceOrchestration Deployment connectors, create a **new link** between the Deployment and the **Servlet** instance.
i. For the **Servlet**, create a **new link** between the **Servlet** interface and the third **Deployment** interface instance. Create a new link between this Deployment instance and the **Server interface**.
j. For the **Servlet**, create a second interface and **link** this to the **Resource connector** interface.
k. Link the **Resource** connector interface to the **ThreadPool** interface**.**

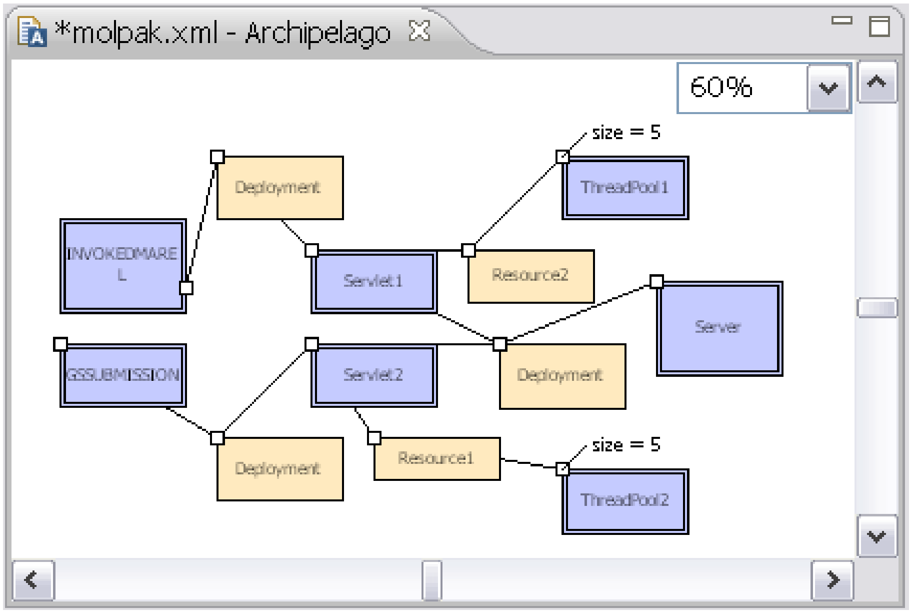
l. Select the **ThreadPool** interface object and right-click, select **Edit Description** and enter "size = 10".

The structure should look like that in the figure below.



4. Create a Dual Servlet Deployment Architecture

a. Select the **Structures** branch, right-click and select **New Structure**.
b. Select the **[New Structure]** item, right-click and edit the description to be **DualServlet.**
c. Repeat steps **a-k** in step 5, however, for this configuration we need: **2x ServiceOrchestration**, **2x ThreadPool**, **2x Servlet**, **1x Server**, **3x Deployment** and **2x Resource.**
d. Link the components and connectors as shown in the figure below.



a. Set each **ThreadPool** interface description to be "size = 5".

3.5.2.2 Create a UML2 Service Deployment Architecture

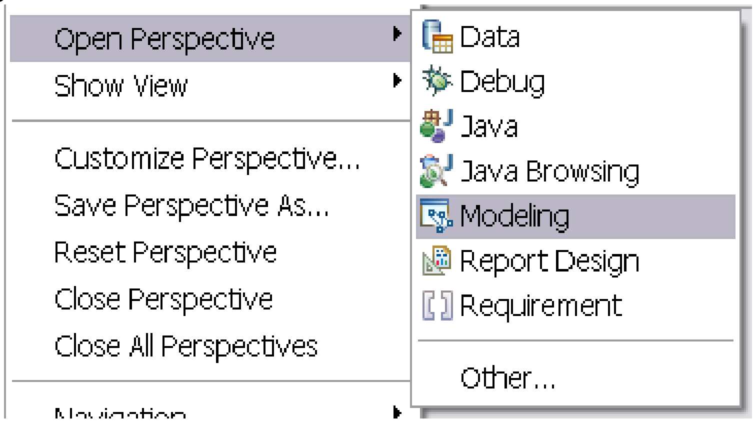
Prerequisites

RSA 7.0

- IBM Rational Software Architect 7 (to create new UML2 Deployment Models)

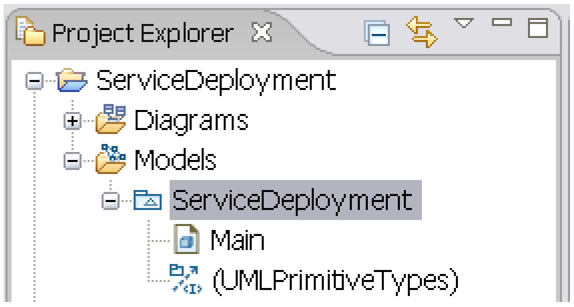
1. Switch to the Modeling Perspective

a. Locate the **Eclipse menu bar**.
b. Select **Window → Open Perspective**.
c. Select **Modeling**.



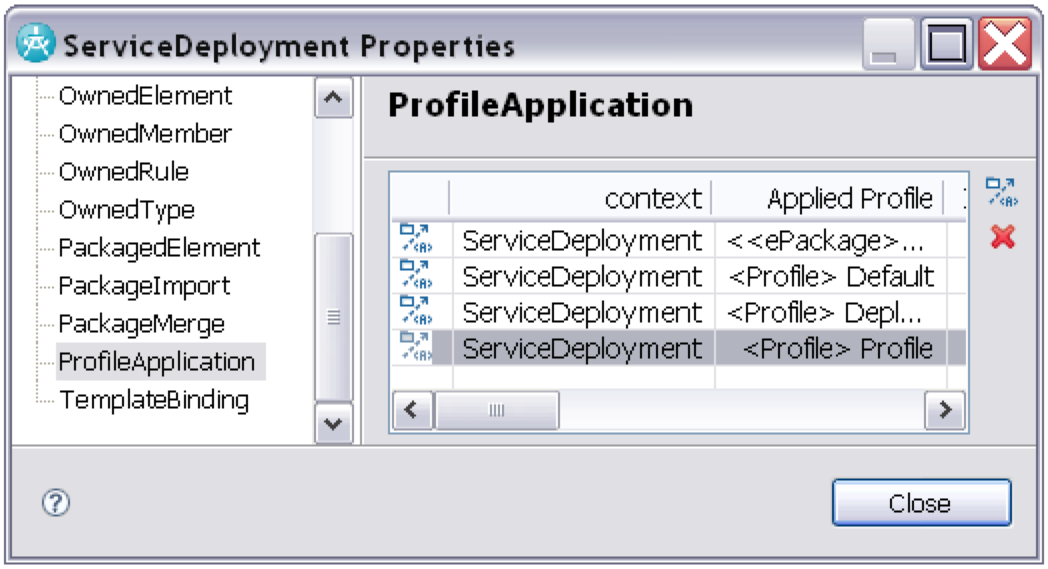
1. Create a new UML Project

a. Select **File → New → Project → UML Project** and click **next.**
b. Enter the name of the project as **ServiceDeployment** and click **next.**
c. Under **Templates** select **Blank Model**.
d. Under **Filename** enter **ServiceDeployment**.
e. Under **Default Diagram type** select **Deployment Diagram**.
f. Click **Finish** to create the new project.



2. Import and Apply the Service Deployment Profile

a. Locate the **ServiceDeployProfile** example profile (or download from the LTSA Samples page).
b. Select **File → Import → General → Existing Projects into Workspace** and click **next.**
c. Under Select root directory, browse to the directory containing the root of the **ServiceDeployProfile,** select the directory and click **ok**.
d. Ensure the **ServiceDeployProfile** is checked under **Projects:** and click **Finish**.
e. In the Navigator View, select the **ServiceDeployment** Project created in Step 1.
f. Under the **Models** branch in the project, select the **ServiceDeployment,** right-click and select **UML Properties** from the pop-up context menu**.**
g. In the **ServiceDeployment** Properties dialog, select the **ProfileApplication** item (on the list to the left).
h. Right-click the ProfileApplication table and select **Insert New Profile Application**.
i. Select the **Profile In Workspace** option and browse to the **ServiceDeployProfile** project imported in a-d. Open the tree and select **Profile.epx** and click **ok** and **ok** again.
j. Select **File → Save** to update the **ServiceDeployment** project.



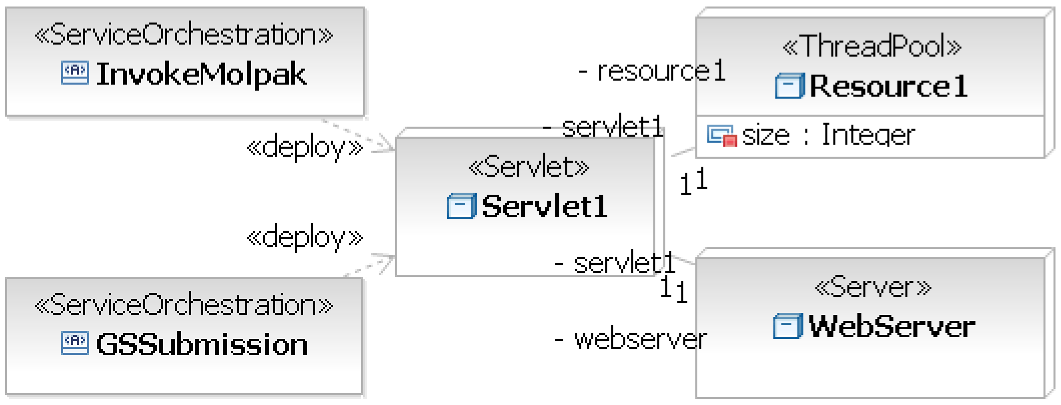
context	Applied Profile
ServiceDeployment	<<ePackage>...
ServiceDeployment	<Profile> Default
ServiceDeployment	<Profile> Depl...
ServiceDeployment	<Profile> Profile

3. Create a Single Servlet Deployment Architecture

a. Under the **ServiceDeployment** Model, select and open the **Main** diagram.
b. Now we need to add some objects to this diagram. Locate the **Palette toolbar** to the right of the main diagram window and select the **Node drop-down list** and select **Stereotyped Node**.
c. Place the new Node on the diagram and select **Create <<Servlet>> Node**.
d. Name the new node object, **Servlet1**.
e. Now create a new Stereotyped Node, and select its type as **<<Server>>**.
f. Repeat the process for a new Stereotyped Node of type **<<ThreadPool>>**. Select the new ThreadPool node and in the **Properties view** add an **attribute** named "**size**" of **<Primitive Type> Integer**, and **Default Value 10**.
g. Now add two **Stereotyped Artifacts,** one for each BPEL orchestration. Select the type as **<<ServiceOrchestration>>**.
h. Now we need to link the objects. Locate the **Palette toolbar** to the right of the main diagram window and select the **Manifestation drop-down list** and select **Deploy**. Select the first ServiceOrchestration node and link it to the Servlet1 node. Repeat linking the second Service Orchestration node to the Servlet1 node.
i. To link the other objects, select the **Association** relationship in the Palette and link **Servlet1** to the

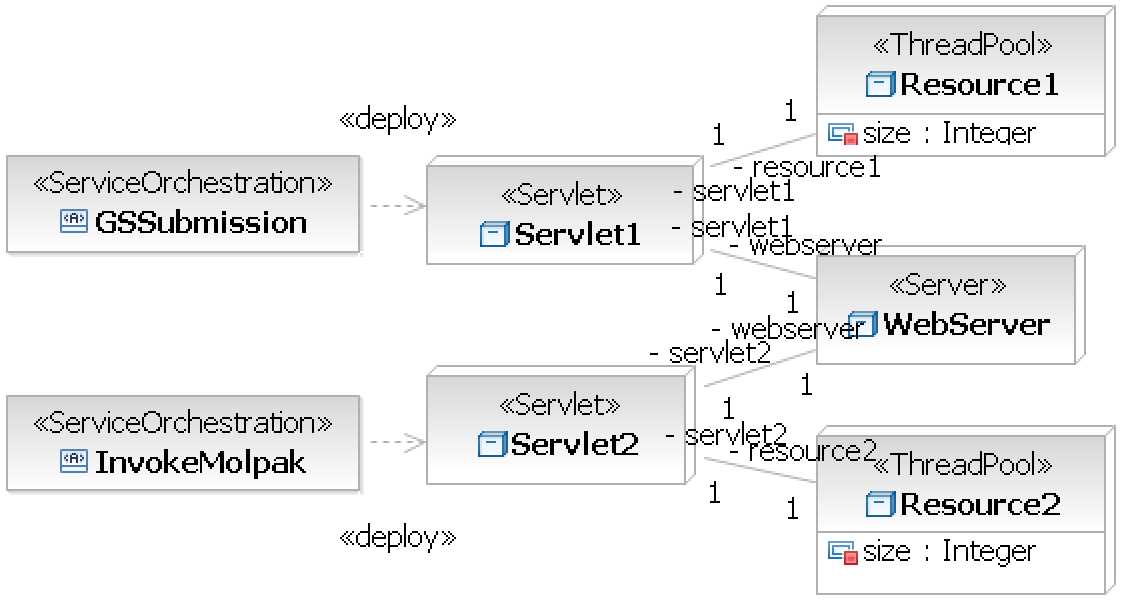
ThreadPool node and **Server** node.

j. Select **File** → **Save** to update the **ServiceDeployment** project.



4. Create a Dual Servlet Deployment Architecture

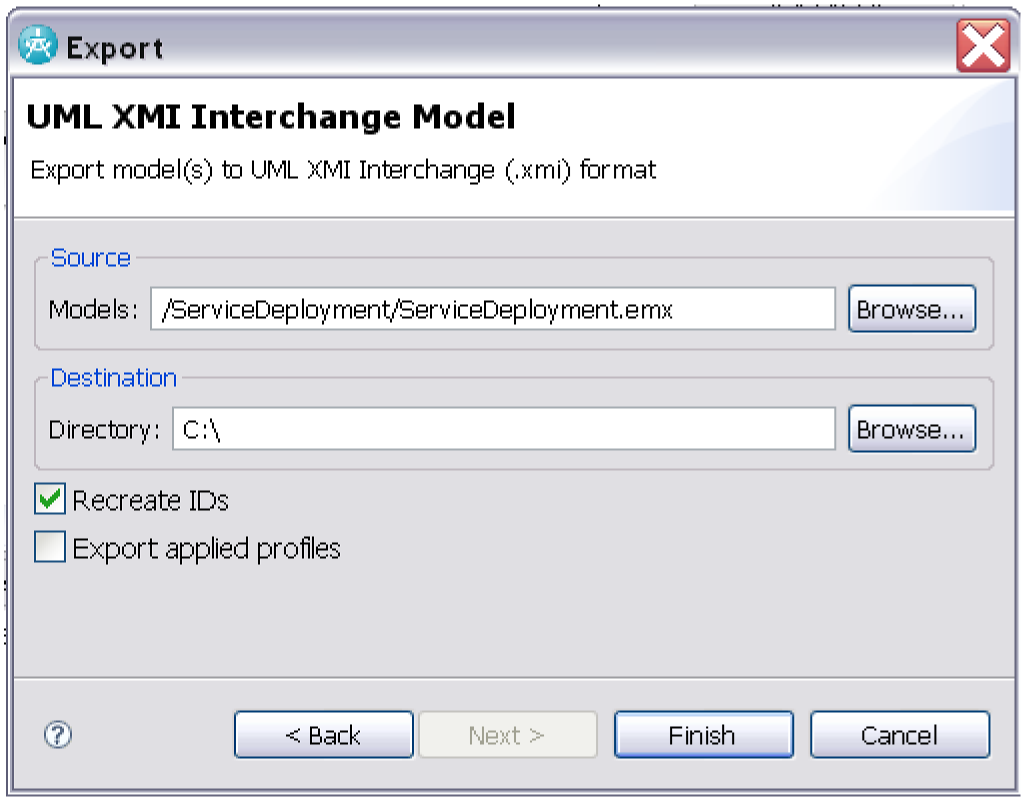
a. Under the **ServiceDeployment project,** select **Models,** right-click and select **Create New Model.**
b. Click **next** and under **filename** enter **ServiceDeployment2.** Click **Finish** to create a new model.
c. Now repeat step 3. but add two <<Servlet>> and two <<ThreadPool>> stereotyped nodes.
d. Link one orchestration to each Servlet and each Servlet to a ThreadPool.
e. Finally link each Servlet to the one WebServer.
a. Select **File** → **Save** to update the **ServiceDeployment** project.



5. Export the Models for use with WS-Engineer

a. Select **File** → **Export** and choose **Other** → **UML XMI Interchange Model** and click **next**.
b. Under **Source Models** browse and select the ServiceDeployment project, and select the **ServiceDeployment.emx** model.
c. Under **Destination Directory** select a suitable output folder.

d. **Check Recreate IDs and Export applied profiles**.
e. Click Finish to export the model in XMI XML format.
f. Repeat a-e for the ServiceDeployment2 model in the same project.
g. Now both of these exported models can be used as input to the Deployment Checking example in **Tutorial 1**.



Export

UML XMI Interchange Model

Export model(s) to UML XMI Interchange (.xmi) format

Source

Models: /ServiceDeployment/ServiceDeployment.emx

Destination

Directory: C:\

☑ Recreate IDs

☐ Export applied profiles

3.5.3 Tutorial: Deployment Analysis

In this tutorial the user will be able to model-check that WS-BPEL orchestrations are "safe" given their behaviour and resource constraints specified in a deployment architecture configuration.

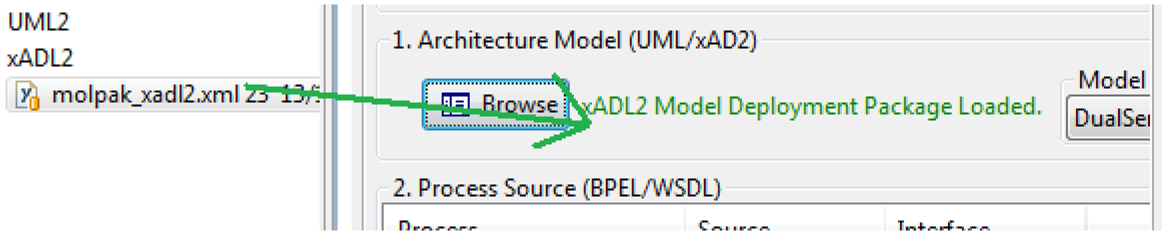
Note: Ensure you have carried out the pre-requisites as described in the Preparation section.

Steps

1. Select the Deployment **Tab on the WS-Engineer View.**

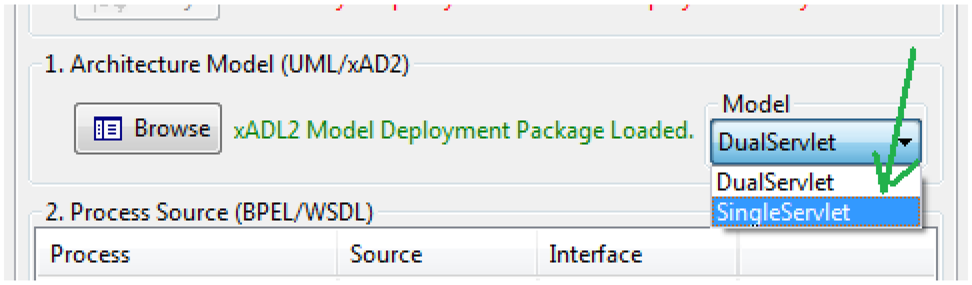
2. Select the Deployment Specification File

Click on the Browse button to select a Deployment Model Package, alternatively you can "drag and drop" the a XADL2.0 xml file or UML2.0 XMI file on the Deployment View as shown below.


UML2
xADL2
molpak_xadl2.xml
1. Architecture Model (UML/xAD2)
Browse
xADL2 Model Deployment Package Loaded.
Model
DualSe
2. Process Source (BPEL/WSDL)

3. Select the Deployment Model

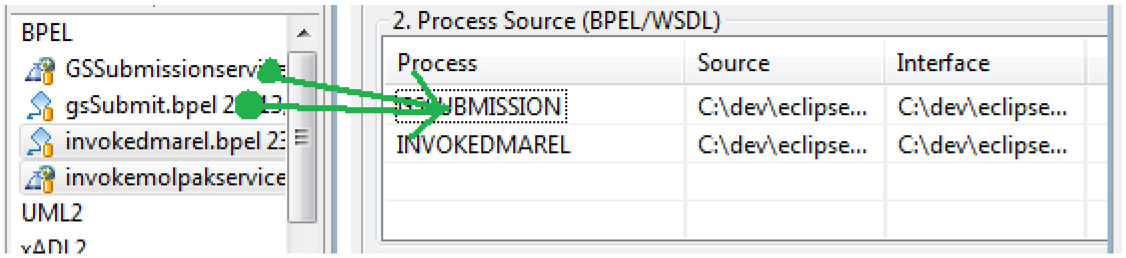
Expanding the Model list aside the Architecture Model will list all available models in the deployment model package.
Note: The **default** model is the first model found in the package.


1. Architecture Model (UML/xAD2)
Browse
xADL2 Model Deployment Package Loaded.
Model
DualServlet
DualServlet
SingleServlet
2. Process Source (BPEL/WSDL)
Process
Source
Interface

4. Select the WS-BPEL and WSDL process pairs to analyse

Drag and drop selected WS-BPEL and WSDL files (together) in the Process Source list box of the deployment view. Repeat this for as many interacting processes as you wish to analyse.

Note: Each WS-BPEL process is paired with one or more WSDL files. In this sample, we associate one WS-BPEL with one WSDL file.

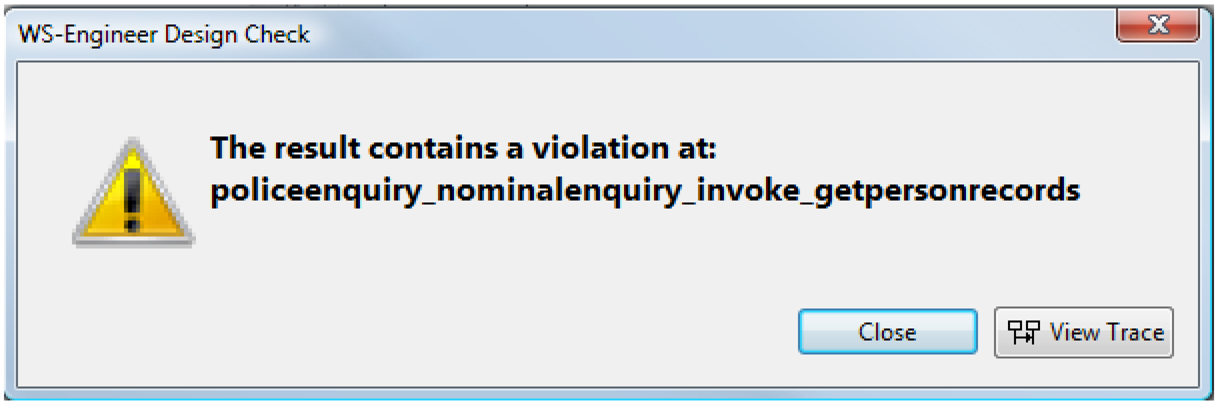


Process	Source	Interface
GSSUBMISSION	C:\dev\eclipse...	C:\dev\eclipse...
INVOKEDMAREL	C:\dev\eclipse...	C:\dev\eclipse...

5. Run the Verify action

When both deployment model and WS-BPEL files are correctly loaded, the Verify button will be enabled. Click the button to start the verification.

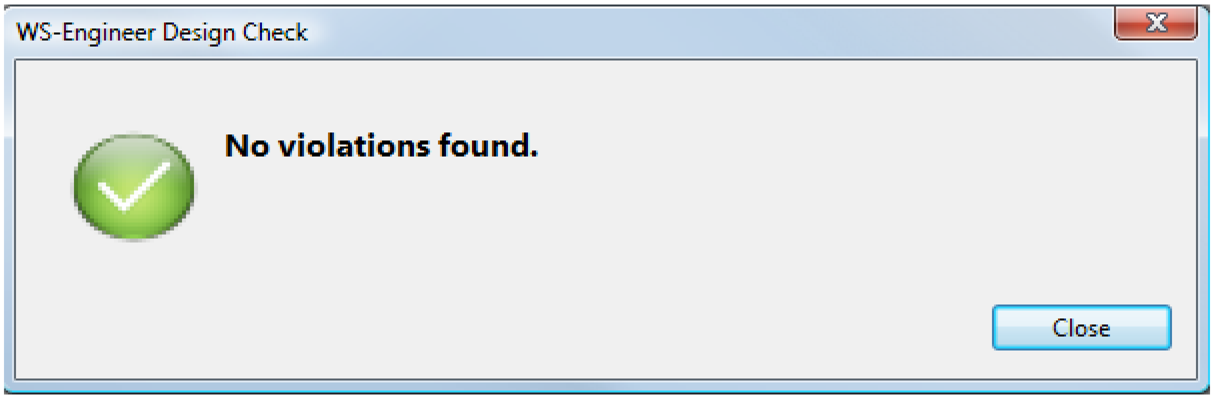
If there is a violation of the behaviour specified in the WS-BPEL and the deployment configuration resources then the point at which the violation occurs is shown as a warning.



The result contains a violation at:
policeenquiry_nominalenquiry_invoke_getpersonrecords

You can click the View Trace button to show the trace to violation as an MSC.

If there are no violations found then **No Violations found** will be shown.



No violations found.

6. Repeat the Verify action for alternative deployment models

END OF TUTORIAL

3.5.4 Further Reading

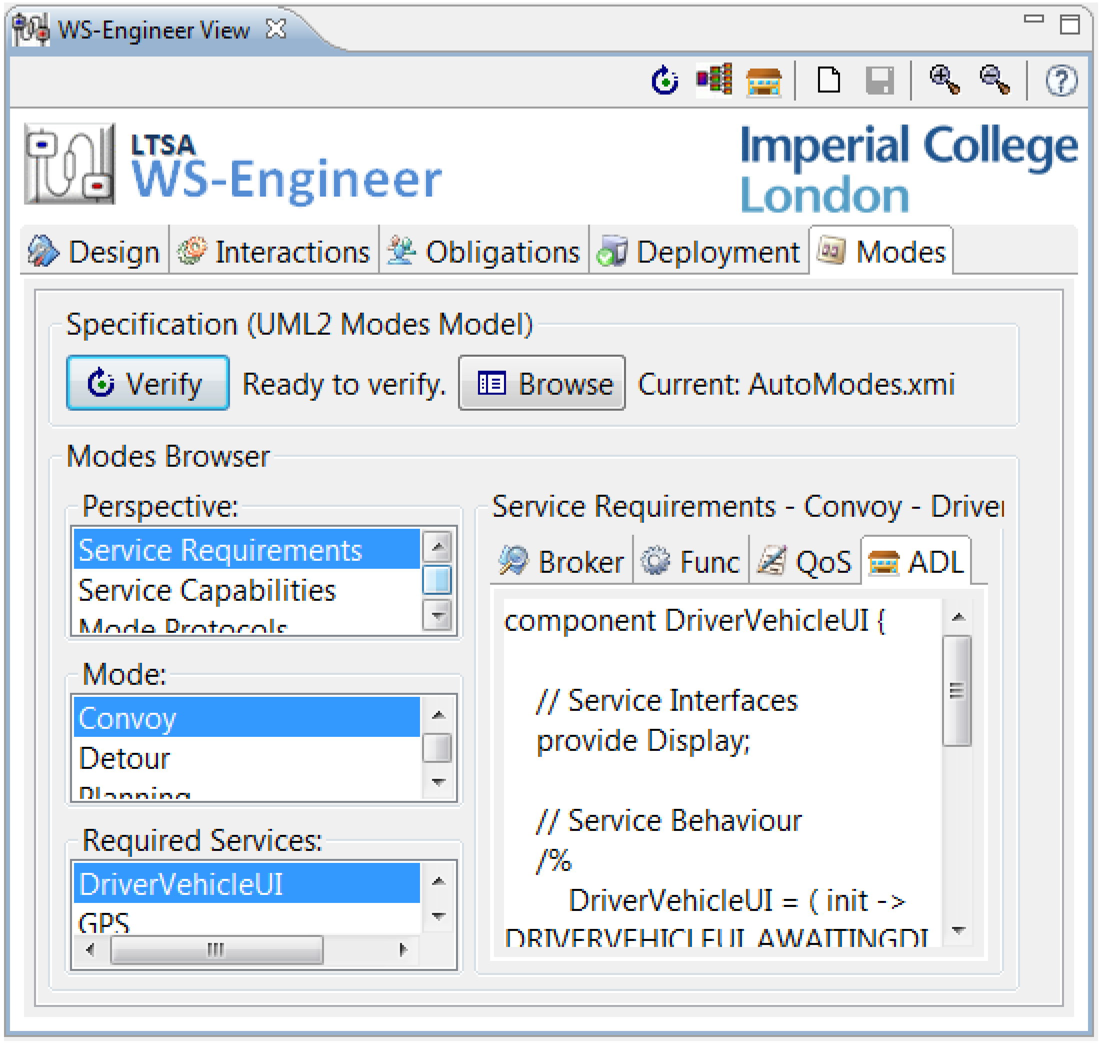
Read the following papers on the techniques used for obligations analysis of web service compositions.

- H.Foster, W. Emmerich, J.Kramer, J.Magee, D.Rosenblum and S.Uchitel, Model Checking Service Compositions under Resource Constraints, in Proceedings of ESEC/FSE 2007, Dubrovnik, Croatia. Sept 2007.
- H.Foster A Rigorous Approach to Engineering Web Service Compositions, PhD Thesis, Imperial College London, January, 2006.

3.6 Modes Analysis

3.6.1 Overview

The modes view of WS-Engineer provides verification of service modes architecture specifications. A service mode architecture consists of a series of mode service collaborations. Typically this will be constructed in UML2 (with a Service Modes profile applied). UML2 Component Structure, Constraints, Activity, Sequence and State Machine Diagrams are translated to formal models and analysed for consistency and correctness.



```
component DriverVehicleUI {

  // Service Interfaces
  provide Display;

  // Service Behaviour
  /%
     DriverVehicleUI = ( init ->
DRIVERVEHICLEUI_AWAITINGDI
```

You can read more about this type of verification in the following publication(s):

H.Foster, S.Uchitel, J.Kramer, J.Magee, Towards Self-Management in Service-oriented Computing with Modes , in Proceedings of the Workshop on Engineering Service-Oriented Applications (WESOA07), Vienna, Austria. Sept 2007.

H.Foster, A.Mukhija, S.Uchitel and D.Rosenblum, A Model-Driven Approach to Dynamic and Adaptive Service Brokering using Modes, in Proceedings of the 6th International Conference on Service Oriented

Computing (ICSOC 2008), Sydney, Australia. December 2008.

3.6.2 Preparation

For Service Modes Analysis you need to have met several pre-requisites:

1. Created a Service Mode Model Project.
2. Created one or more Service mode Collaborations
3. Exported the model in UML2 XMI Model format.

The sections of preparation illustrate how to create these pre-requisites.

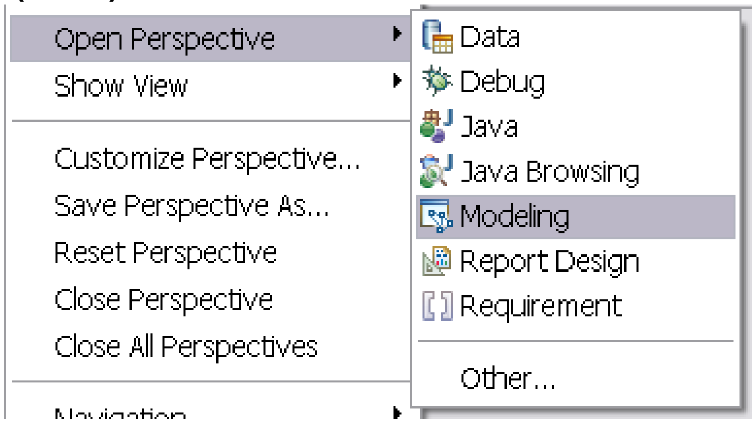
3.6.2.1 Create a Service Mode Model Project

This is an example of how to create a Service Mode Architecture Model Project. We will use the Automotive Case Study of the SENSORIA project to illustrate Service Modes and adaptation requirements.

Note: This example uses the IBM Rational Software Architect 7.5 Standard Edition tool to create a Service Mode Architecture specification with applied Service Modes Architectures Profile.

1. In RSA 7.5, switch to the Modeling Perspective

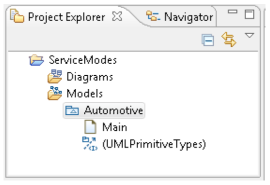
a. Locate the **Eclipse menu bar**.
b. Select **Window → Open Perspective → Other**.
c. Select **Modeling (default)**.



2. Create a new UML Model Project

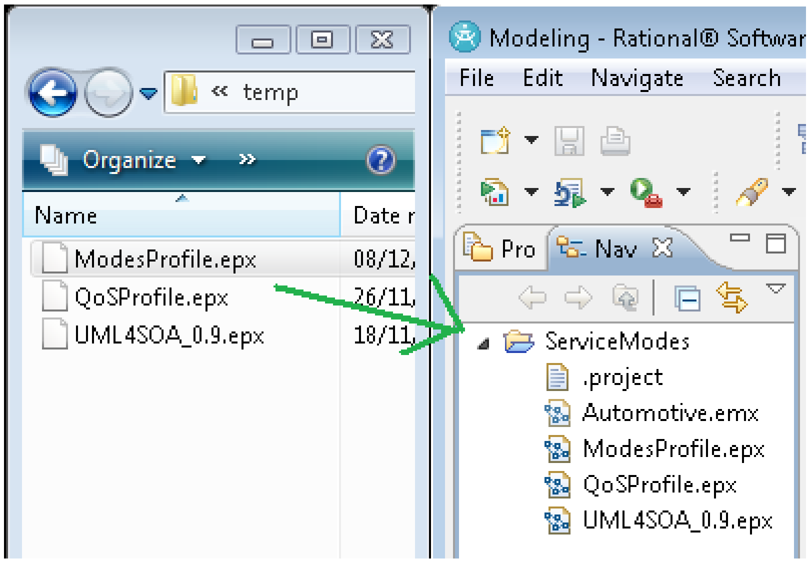
a. Select **File → New → Other → Modeling → Model Project** and click **next.**
b. Enter the name for the project as **ServiceModes** and click **next.**
c. Keep new model as "**Standard Template**" and click **next.**
d. Under **Categories** select **General.**
e. Under **Templates** select **Blank Package**.
f. Under the model **Filename** enter **Automotive**.
g. Click **next** to continue.
h. Select Package Type as **Model.**
i. Keep the Diagram Type as **Freeform Diagram.**

j. Click **Finish** to create the new project.
k. Notice a new project is visible in the Project Explorer (as show below).

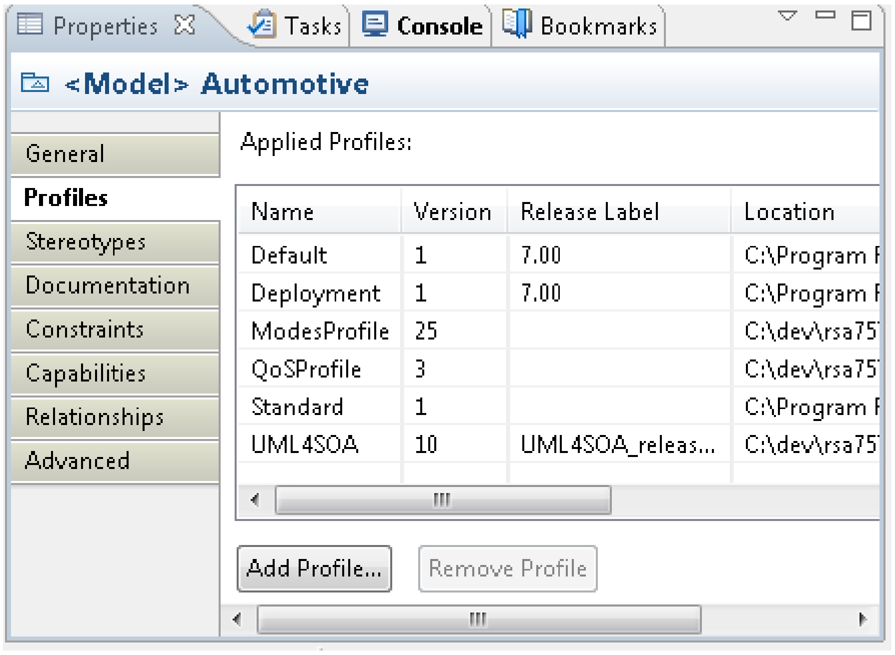


3. Import and apply the Service Modes Architecture profiles

a. Switch to the **Navigator view**
b. From Windows Explorer, locate the Service Modes Architecture profile files and drag-and-drop them on to the **ServiceModes** project folder in the Navigator view (as shown below)



c. Switch back to the **Project Explorer view** and select the **Automotive** model branch
d. Locate the model **Properties** view and select **Profiles.**
e. On the view select **Add Profile** → **Profile in Workspace** →**Browse** and select the **ModesProfile.epx** under **ServiceModes.**
f. **Repeat e.** adding both QoSProfile.epx and UML4SOA_0.9.epx

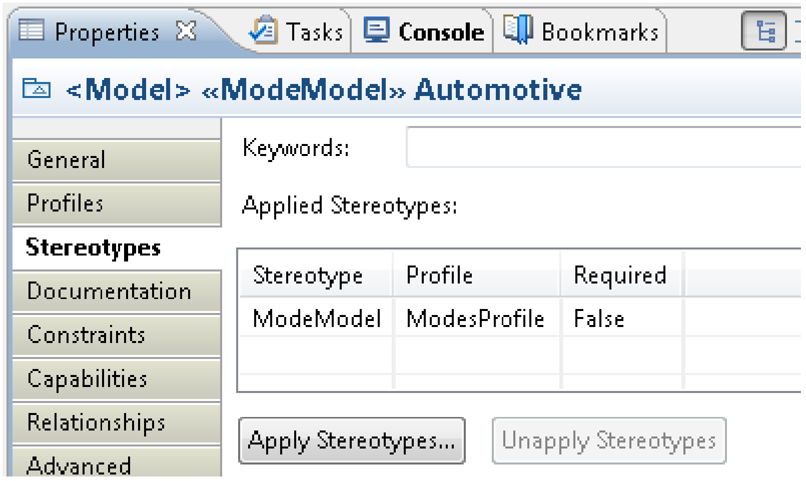


<Model> Automotive

Applied Profiles:

Name	Version	Release Label	Location
Default	1	7.00	C:\Program F
Deployment	1	7.00	C:\Program F
ModesProfile	25		C:\dev\rsa75
QoSProfile	3		C:\dev\rsa75
Standard	1		C:\Program F
UML4SOA	10	UML4SOA_releas...	C:\dev\rsa75

g. Switch to the **Stereotypes** perspective of the model properties.
h. Select **Apply Stereotypes** select the **ModesModel** from the **ModesProfile** profile and click **OK.**
i. Note that the stereotype is added to the applied stereotypes list.



<Model> «ModeModel» Automotive

Keywords:

Applied Stereotypes:

Stereotype	Profile	Required
ModeModel	ModesProfile	False

4. Save the changes

a. Locate the **Eclipse menu bar**.
b. Select **File → Save All**.

3.6.2.2 Create A Service Mode Collaboration

This is an example of how to create a Service Mode Architecture Model Collaboration. We will use the Automotive Case Study **of the SENSORIA project to illustrate Service Modes and adaptation requirements.**

Please follow the instructions to create a new Service Mode Architecture Model Project if not already completed.

Note: This example uses the IBM Rational Software Architect 7.5 Standard Edition tool to create a Service Mode Architecture specification with applied Service Modes Architectures Profile.

1. Add a Mode Package

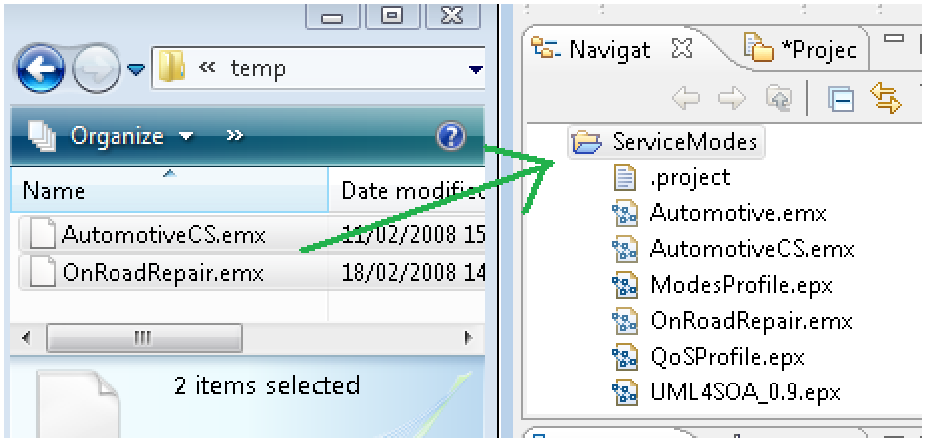
a. In **Project Explorer,** select the **ModeModel** model (e.g. **Automotive**) and right-click to show the context pop-up menu.
b. Select **Add UML → Package.** A new package branch will be added to the ModeModel in the Project Explorer view.
c. Rename the new package **Convoy.**
d. Switch to the **Stereotypes** perspective of the model properties.
e. Select **Apply Stereotypes** select the **Mode** from the **ModesProfile** profile and click **OK.**
f. Note that the stereotype is added to the applied stereotypes list.

2. Add a Mode Collaboration

a. In **Project Explorer,** select the Mode **Convoy** and right-click to show the context pop-up menu.
b. Select **Add UML → Collaboration.** A new collaboration branch will be added to the Convoy Mode in the Project Explorer view.
c. Rename the new package **ConvoyCollaboration.**
d. Switch to the **Stereotypes** perspective of the model properties.
e. Select **Apply Stereotypes** select the **ModeCollaboration** from the **ModesProfile** profile and click **OK.**
f. Note that the stereotype is added to the applied stereotypes list.
g. In **Project Explorer,** select the ModeCollaboration **ConvoyCollaboration** and right-click to show the context pop-up menu.
h. Select **Add Diagram → Composite Structure Diagram.** A new component diagram branch will be added to the ConvoyCollaboration node in the Project Explorer view.
i. Rename the new package **ConvoyComposite.**

3. Import some Components

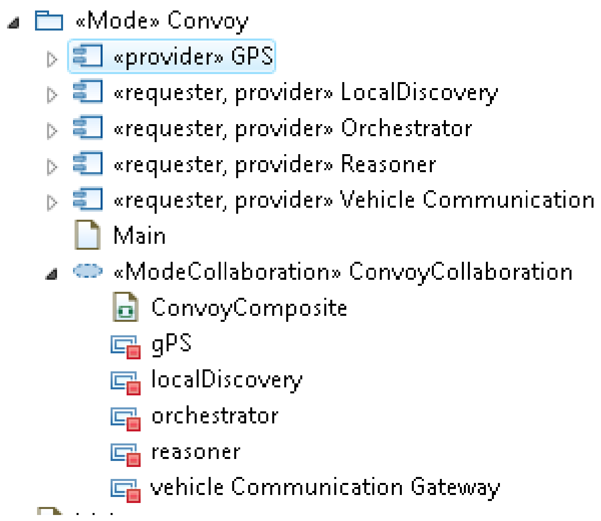
a. Switch to the **Navigator view**
b. From Windows Explorer, locate the **Automotive Case Study** files and drag-and-drop them on to the **ServiceModes** project folder in the Navigator view (as shown below)



c. Switch to the **Project Explorer** view
d. Open the **OnRoadAssistance** model and navigate to the **Vehicle** package.
e. Select **GPS**, **LocalDiscovery**, **Orchestrator**, **Reasoner** and **Vehicle Communication Gateway** components, right-click and select **Copy**.
f. Navigate back to the Mode **Convoy** package and right-click and select **Paste**.
g. Then back in the OnRoadAssistance model, navigate to the VehicleExternal package and select the **RemoteDiscovery** component.
h. Copy and paste the RemoteDiscovery component in the Mode Convoy package.
i. Now select the Mode Convoy package, and right-click and select **Add UML → Component**. Name the new component **OtherVehicle.**
j. For each component we need to add stereotypes for their service role. We add this by selecting each component, and in the **Properties** view adding a **stereotype** based upon their role. To add a keyword stereotype, select the component, select Stereotypes and enter the stereotype in the keywords text box.
k. Stereotype the components added as follows:

GPS	provider
LocalDiscovery	requester, provider
Orchestrator	requester, provider
Reasoner	requester, provider
Vehicle Communication Gateway	requester, provider
RemoteDiscovery	requester, provider
OtherVehicle	provider

l. Drag and drop each component on to the **ConvoyComposite** diagram editor.
m. Note now you have some new properties of the ConvoyComposite (for each component added) as shown below.


«Mode» Convoy
«provider» GPS
«requester, provider» LocalDiscovery
«requester, provider» Orchestrator
«requester, provider» Reasoner
«requester, provider» Vehicle Communication
Main
«ModeCollaboration» ConvoyCollaboration
ConvoyComposite
gPS
localDiscovery
orchestrator
reasoner
vehicle Communication Gateway

n. For visual clarity, add the same stereotypes to each component property.

3. Linking Required and Provided Services

a. On the **ConvoyComposite** diagram locate the Palette sidebar and under **Composite Structure** drop the list for Provided Interface to select a Required Interface.
b. Then move over to the **LocalDiscovery** component and locate the **Port** <<service>> localDiscoverServices. Drop the Required Interface on this port.
c. When prompted, choose **Select Existing Element** and type **Location** in select an element text box. Choose the **«serviceInterface» Location - Automotive::Convoy::GPS::Location** element**.**
d. Now go back to the Palette sidebar and select **Connector.** Move back to the LocalDiscovery component and click the port then whilst holding down the left mouse button (to drag the connecting line) move down to the **GPS** component and release the connector on the visible port. This binds the two interfaces together. Name the connector **GPSBinding**.
e. Now proceed to add Ports, Required and Provided Interfaces to the rest of the Composite Structure as listed in the table below.

From Service	**To Service**	**To Provided Interface (new = new port)**	**Connector Name**
Orchestrator	LocalDiscoveryService	findLocal (new)	localBind
Orchestrator	Reasoner	ServiceSelection	reasonserBind
Orchestrator	Vehicle Communication Gateway	AccessExtServices (new)	vcgBind
Vehicle Communication Gateway	RemoteDiscovery	RemoteServices	remoteBind
RemoteDiscovery	OtherVehicle	VCG (new)	vcgBinding

4. Create other Mode Packages for "Detour" and "Planning"

a. Now repeat steps 1-3 but create one Mode Package for a "Detour" mode composition and one for "Planning".
b. The differences in each package are as follows:
c. For the Detour package, replace the OtherVehicle component with a new "HWEmergency" component. Add suitable ports, interfaces and bindings between the Vehicle Communication

Gateway and the HWEmergency component.

d. For the Planning package, replace the OtherVehicle component with a new "RoutePlanner" component. Add suitable ports, interfaces and bindings between the Vehicle Communication Gateway and the RoutePlanner component.

3.6.2.3 Exporting a Modes Package

This is an example of how to export a UML2 Service Mode Architecture Model to UML XMI format We use the model created in Create a Service Mode Collaboration section.

Export

a. In **Project Explorer,** select the **<<Mode Model>> Automotive Model** and right-click and select **New Folder.** Name the folder **exported.**
b. Re-select the **<<Mode Model>> Automotive Model** and right-click and select **Export.**
c. Select **UML 2.1 XMI Interchange Model** and click **next**.
d. Select destination as **Workspace** and browse to the new **exported** folder under the Model.
e. Enable both **Recreate IDs** and **Export applied profiles**.
f. Click **Finish** to export the model to xmi files.

3.6.3 Tutorial: Modes Analysis

In this tutorial the user will be able to model-check that Service Mode Architectures are "safe" given their architecture, behaviour and constraints in a service modes architecture configuration.

Note: Ensure you have carried out the pre-requisites as described in the Preparation section.

THIS TUTORIAL IS STILL UNDER CONSTRUCTION

Steps

1. **Select the** Modes **Tab on the WS-Engineer View.**

2. **Select the Service Modes Specification File**

Click on the Browse button to select a Service Modes Model Package, alternatively you can "drag and drop" the a UML2 XMI file on the Modes View as shown below. See Exporting a Modes Package for a sample of how to export a model to UML XMI format.

3.6.4 Further Reading

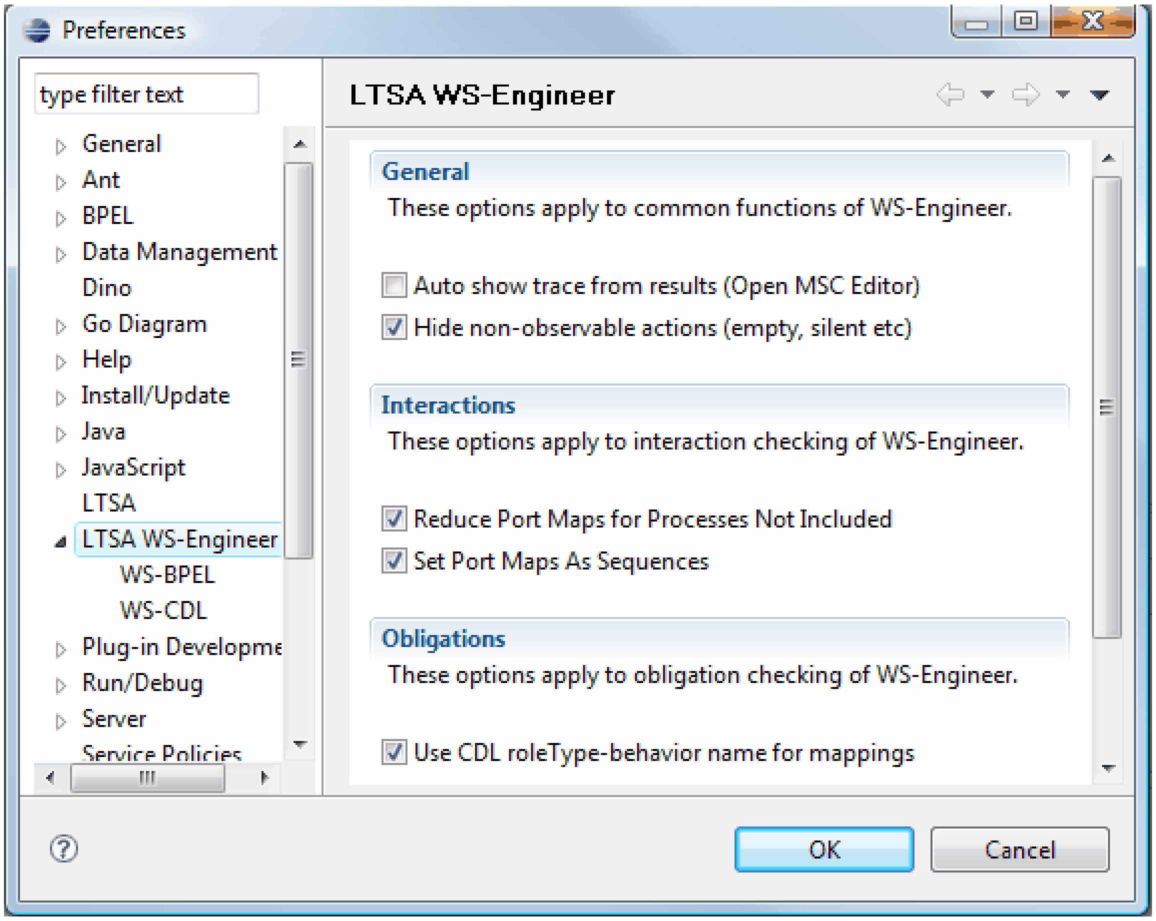
You can read more about this type of verification in the following publication(s):

H.Foster, S.Uchitel, J.Kramer, J.Magee, Towards Self-Management in Service-oriented Computing with Modes , in Proceedings of the Workshop on Engineering Service-Oriented Applications (WESOA07), Vienna, Austria. Sept 2007.

H.Foster, A.Mukhija, S.Uchitel and D.Rosenblum, A Model-Driven Approach to Dynamic and Adaptive Service Brokering using Modes, in Proceedings of the 6th International Conference on Service Oriented Computing (ICSOC 2008), Sydney, Australia. December 2008.

4 Preferences

WS-Engineer includes a series of preferences pages, accessed from the Eclipse menu item: **Window → Preferences → LTSA WS-Engineer.**



LTSA WS-Engineer

General

These options apply to common functions of WS-Engineer.

- [ ] Auto show trace from results (Open MSC Editor)
- [✓] Hide non-observable actions (empty, silent etc)

Interactions

These options apply to interaction checking of WS-Engineer.

- [✓] Reduce Port Maps for Processes Not Included
- [✓] Set Port Maps As Sequences

Obligations

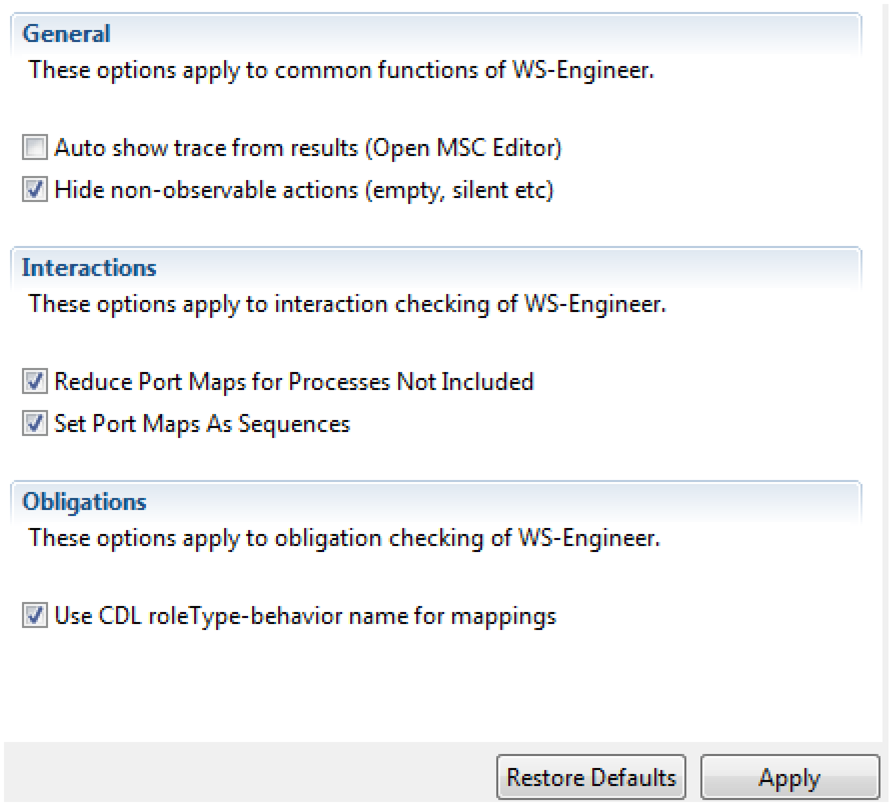
These options apply to obligation checking of WS-Engineer.

- [✓] Use CDL roleType-behavior name for mappings

Preferences are split in to General WS-Engineer verification preferences (applies to actions taken in the WS-Engineer view), WS-BPEL preferences (applies to the translation of WS-BPEL processes in the Eclipse environment and LTSA-WSBPEL Editor) and WS-CDL preferences (applies to the translation of WS-CDL choreography specifications).

4.1 WS-Engineer Preferences

The WS-Engineer preferences page is accessed from the Eclipse menu item: **Window → Preferences → LTSA WS-Engineer.**


General
These options apply to common functions of WS-Engineer.

- [ ] Auto show trace from results (Open MSC Editor)
- [✓] Hide non-observable actions (empty, silent etc)

Interactions
These options apply to interaction checking of WS-Engineer.

- [✓] Reduce Port Maps for Processes Not Included
- [✓] Set Port Maps As Sequences

Obligations
These options apply to obligation checking of WS-Engineer.

- [✓] Use CDL roleType-behavior name for mappings

General

Option	Description
Auto show trace from results (Open MSC Editor)	This option specifies whether the default action of verification is to open a message sequence trace in the LTSA-MSC editor. If **checked**, the editor will be opened. If **unchecked**, a message box will open and then the user is given a choice of whether to show the trace.
Hide non-observable actions (empty, silent etc)	This option specifies whether WS-Engineer hides any non-observable actions in service process (WS-BPEL) or choreography descriptions. If **checked,** the actions are hidden**.** If unchecked, the actions are observable in verification.

Interactions

Option	Description
Reduce Port Maps for Processes Not Included	This option specifies that any partner process port mappings not found in interaction checking are reduced to the observable actions of the client process only. If **checked**, the interaction checking will reduce the ports. If **unchecked**, the

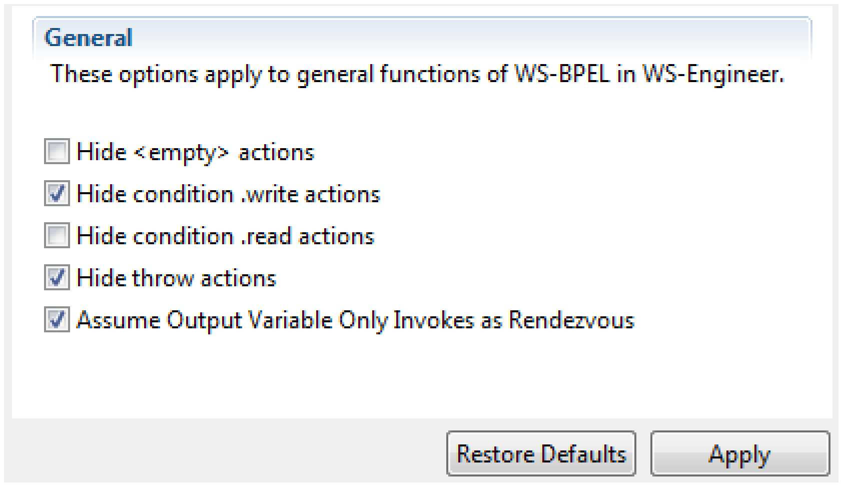
Option	Description
	full port mapping (invoke, receive, reply) will be created.
Set Port Maps as Sequences	This option specifies whether WS-Engineer builds a port mapping as a parallel composition or as a sequence of actions. If **checked,** the actions are built in sequence**.** If **unchecked**, the actions are built as a parallel composition.

Obligations

Option	Description
Use CDL roleType-behaviour name for mappings	This options specifies that the CDL roleType-behaviour name is used to link the name of a partner in a CDL choreography specification with a WS-BPEL partinerLink role name. If **checked**, the CDL roletype is used. If **unchecked**, the WS-BPEL process name is used to link between CDL and WS-BPEL.

4.2 WS-BPEL Preferences

The WS-Engineer WS-BPEL preferences page is accessed from the Eclipse menu item: **Window → Preferences → LTSA WS-Engineer → WS-BPEL.**


General

These options apply to general functions of WS-BPEL in WS-Engineer.

- [ ] Hide <empty> actions
- [✓] Hide condition .write actions
- [ ] Hide condition .read actions
- [✓] Hide throw actions
- [✓] Assume Output Variable Only Invokes as Rendezvous

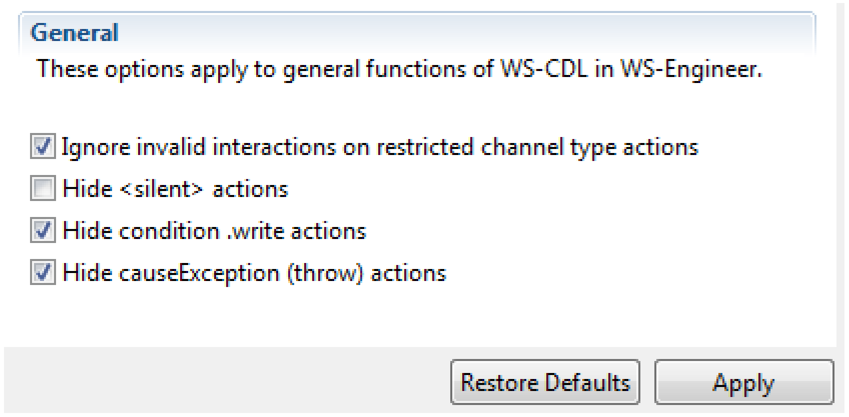
General

Option	Description
Hide <empty> actions	This option specifies whether the empty construct of WS-BPEL is hidden in translation. If **checked**, the empty action is hidden. If **unchecked**, the empty actions are included in the observable actions.

Option	Description
Hide condition .write actions	This option specifies whether the write actions of conditional constructs of WS-BPEL are hidden in translation. If **checked**, the write actions are hidden. If **unchecked**, the actions are included in the observable actions.
Hide condition .read actions	This option specifies whether the read actions of conditional constructs of WS-BPEL are hidden in translation. If **checked**, the read actions are hidden. If **unchecked**, the actions are included in the observable actions.
Hide throw actions	This option specifies whether the throw actions of WS-BPEL are hidden in translation. If **checked**, the throw actions are hidden. If **unchecked**, the throw actions are included in the observable actions.
Assume Output Variable Only Invokes as Rendezvous	This option specifies that if there is only a Output Variable specified in an Invoke action then the default mapping is to build a rendezvous (invoke, invoke-receive) port model. If **checked**, the mapping builds a rendezvous model (invoke→ invoke_receive). If **unchecked**, the mappings builds a asynchronous mapping (invoke→ receive).

4.3 WS-CDL Preferences

The WS-Engineer WS-CDL preferences page is accessed from the Eclipse menu item: **Window → Preferences → LTSA WS-Engineer → WS-CDL.**


General
These options apply to general functions of WS-CDL in WS-Engineer.

[✓] Ignore invalid interactions on restricted channel type actions
[ ] Hide <silent> actions
[✓] Hide condition .write actions
[✓] Hide causeException (throw) actions

General

Option	Description
Hide <silent> actions	This option specifies whether the silent construct of WS-CDL is hidden in translation. If **checked**,

Option	Description
	the silent action is hidden. If **unchecked**, the silent actions are included in the observable actions.
Hide condition .write actions	This option specifies whether the write actions of conditional constructs of WS-CDL are hidden in translation. If **checked**, the write actions are hidden. If **unchecked**, the actions are included in the observable actions.
Hide causeException (throw) actions	This option specifies whether the causeException actions of WS-CDL are hidden in translation. If **checked**, the causeException actions are hidden. If **unchecked**, the causeException actions are included in the observable actions.
Ignore invalid interactions on restricted channel type actions	This option specifies whether WS-Engineer checks the type of interaction used over a channel type and the restrictions imposed on the interactions using the channel. If **checked** and an interaction is specified in WS-CDL which is not valid on the channel type then an error added to the Error view. If **unchecked,** then the action violation is ignored and the model is built as if there was no violation.

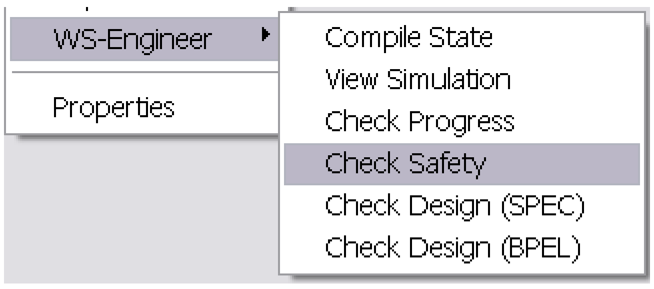
5 Appendix

5.1 Short cuts

Below is a list of actions which can be used for quick access to analysis functions in **WS-Engineer**.

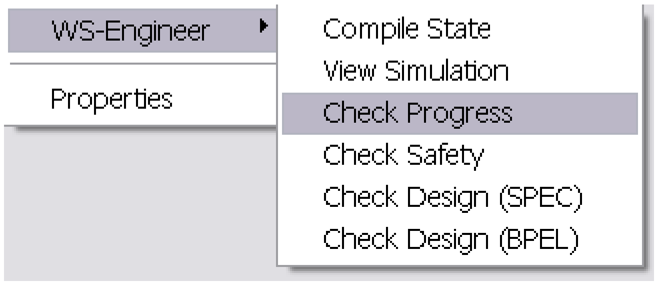
1. **BPEL4WS / WS-BPEL**

Safety Analysis Model Check



a. Select the **BPEL4WS** source file in the **Navigator** View.
b. Right-click the file and select **WS-Engineer → Check Safety.**

Progress (Liveness) Analysis

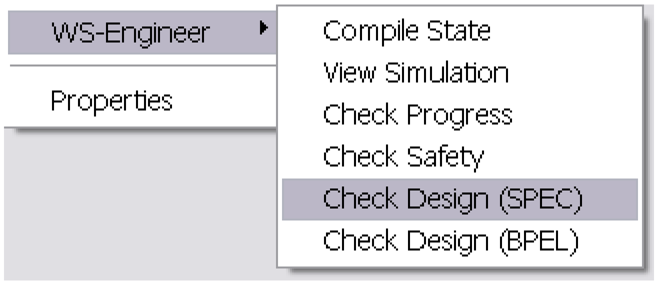


a. Select the **BPEL4WS** source file in the **Navigator** View.
b. Right-click the file and select **WS-Engineer** → **Check Progress.**

Check BPEL Design Against MSC

a. Select the **BPEL4WS** source file in the **Navigator** View.
b. Right-click the file and select **WS-Engineer** → **Check Design (BPEL).**

Check MSC Design Against BPEL



a. Select the **BPEL4WS** source file in the **Navigator** View.
b. Right-click the file and select **WS-Engineer** → **Check Design (SPEC).**

5.2 Glossary and Acronyms

Term	Definition	Link (web/papers/reports)
Scenario	A brief description of events or series of events.	Wikipedia
Implied Scenario	A mis-match between specified behaviour and architecture.	S. Uchitel, J. Kramer and J. Magee. *Synthesis of Behavorial Models from Scenarios*. IEEE Transactions on Software Engineering. Volume 29, Number 2, February 2003.
Negative Scenario	An undesirable implied scenario, from the point of view of a stakeholder.	S. Uchitel, J. Kramer and J. Magee. *Negative Scenarios for Implied Scenario Elicitation*. In 10th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE'02), Charleston, November 2002
MSC	Message Sequence Chart	ITU Recommendation Z-120 04-2004, Message Sequence Charts (MSC)
hMSC	Higher-Level Message Sequence Chart	ITU Recommendation Z-120 04-2004, Message Sequence Charts (MSC)

bMSC	Basic Message Sequence Chart ITU Recommendation Z-120 04-2004, Message Sequence Charts (MSC)

Acronym	**Full Title**
FSP	Finite State Processes
LTSA	The Labelled Transition System Analyser
MSC	Message Sequence Chart
BPEL4WS	Business Execution Language for Web Services (v1.0/1.1)
WS-BPEL	The Web Services Business Execution Language (BPEL4WS v2.0)
WS-CDL	The Web Services Choreography Description Language
WSDL	The Web Services Description Language (WSDL)

Index

END OF GUIDE