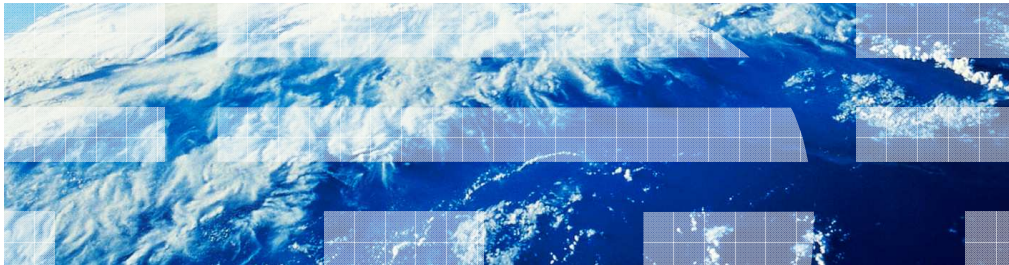


WebSphere Enterprise Service Bus service gateway in conjunction with WebSphere Service Registry and Repository



This presentation provides an overview of WebSphere® Enterprise Service Bus service gateway pattern in conjunction with WebSphere Service Registry and Repository.

Table of contents

- Introduction
- Static versus dynamic service gateway
- Service gateway Interface
- Dynamic service gateway implementation
 - Walkthrough of example scenario
 - Configure router (endpoint look up) primitive
 - WebSphere Service Registry and Repository definition
- Running the example application
- Summary

The goals of this presentation are to provide you an introduction about WebSphere Enterprise Service Bus service gateway, describe the differences between static and dynamic service gateway and explore the functionality of dynamic service gateway in conjunction with WebSphere Service Registry and Repository. This presentation explains different aspects of dynamic service gateway implementation starting with an introduction to service gateway interface, configuration of router primitive followed by creation of WebSphere Service Registry and Repository definitions through the administrative console. Different aspects of service gateway implementation are explained to you with an example application.

Introduction: Service gateway offerings

- Single point of access
- Handles messages for different port types
- Allows common operations on messages
- Allows the message header or body or both manipulated
- Dynamic end point selection
- Supports these bindings:
 - Web service
 - HTTP
 - JMS, Generic JMS, MQ JMS, MQ

The service gateway is a pattern that acts as a proxy to a variety of services. It provides a single point of access to the services, handles messages of different port types, and performs common operations such as logging, monitoring, auditing, and so on. The service gateway interface allows the message header or body to be manipulated depending on the type of gateway in use. The service gateway can be exposed as a Web service, HTTP, JMS, MQ, or MQ JMS binding.

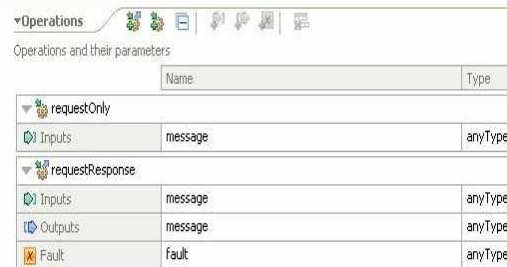
Static versus dynamic service gateway

- *Static service gateway*
 - The endpoint address of the service provider is known before the message enters the gateway
 - Transforms the request from the protocol-specific format to the format matching the service provider interface
 - Service gateway can enrich the transformed request before routing it to the service provider
- *Dynamic service gateway*
 - Service provider endpoint is determined at run time
 - Supports service registry look up and data base look up
 - Does not manipulate the body of the message. Routes the request as-is to the service provider

In static service gateway, service provider endpoint is determined at development time. Each callout node represents a service provider whose endpoint address is defined in the corresponding import. Using static service gateway the request message can be enriched or transformed before routing it to the service provider. As opposed to static service gateway, dynamic service gateway selects the service provider at runtime. Dynamic service gateway supports registry look up, database look up and custom code to select the service end point. A dynamic service gateway does not manipulate the request message and typically routes the requests to the selected service provider as-is.

Service gateway interface

- Single generic gateway interface
- Handle messages for different port types



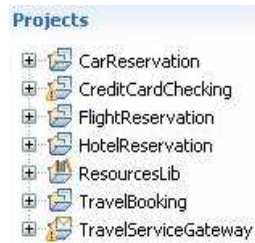
The screenshot shows a table titled "Operations and their parameters" with columns for Name and Type. It lists two main operations: requestOnly and requestResponse. requestOnly has an Input of message (anyType). requestResponse has an Input of message (anyType), an Output of message (anyType), and a Fault of fault (anyType).

	Name	Type
requestOnly		
Inputs	message	anyType
requestResponse		
Inputs	message	anyType
Outputs	message	anyType
Fault	fault	anyType

The service gateway provides a generic Interface that allows the messages for any port type. The requestOnly operation handles one-way operations with parameters of anyType. The requestResponse operation handles two-way operations.

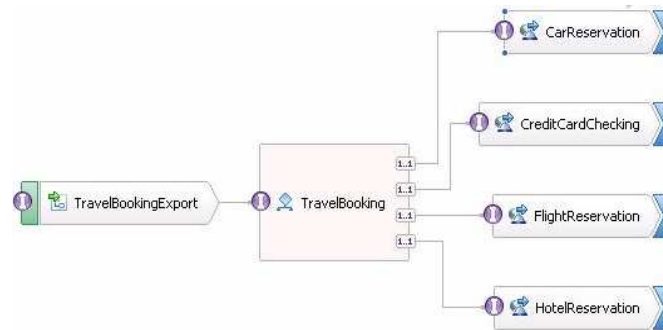
Dynamic service gateway: Example scenario

- Import the accompanied PI in to WebSphere Integration Developer 6.2
- Application should contain these modules:



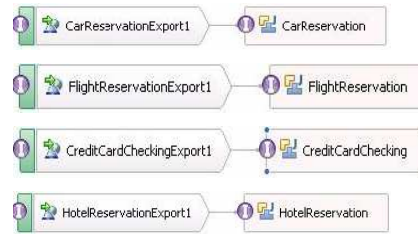
The next few slides explain the implementation of dynamic service gateway with the help of an example scenario. This requires you to download the accompanying project interchange and import the application into WebSphere Integration Developer V6.2. You will see the different modules of the application and learn the procedure to create the dynamic service gateway.

Client module: TravelBooking



TravelBooking is a short-running business process that orchestrates the process of travel reservation. Customers can submit data about the travel arrangements they want to make and receive a confirmation number when the travel arrangements are made successfully. The business flow involves the invocation of four different Web services.

Service endpoints (1 of 2)



Export: CarReservationExport1 (Web Service Binding)

Description	Transport: SOAP1.1/HTTP
Details	Address: <u>http://localhost:9089/CarReservationWeb/sca/CarReservationExport1</u>
Qualifiers	
Binding	

This slide shows the end points of four different services and their actual addresses. With the introduction of the service registry, information about the back-end services is now stored in the registry repository and maintained there. The registry saves all the static information of the back-end services, including the endpoint URL and the parameters. The dynamic service gateway makes use of the service registry and selects the service provider at runtime.

Service endpoints (2 of 2)

Import: CarReservation (Web Service Binding)

Description	Transport: SOAP1.1 HTTP
Details	Address: <u>http://localhost:9089/TravelServiceGatewayWeb/sca/TravelServiceGatewayExport</u>
Qualifiers	
Binding	Port: CarReservationExport1_CarReservationHttpPort
Policy Sets	
JAX-WS Handlers	Service: CarReservationExport1_CarReservationHttpService
Propagation	Namespace: http://ResourcesLib/CarReservation/Binding

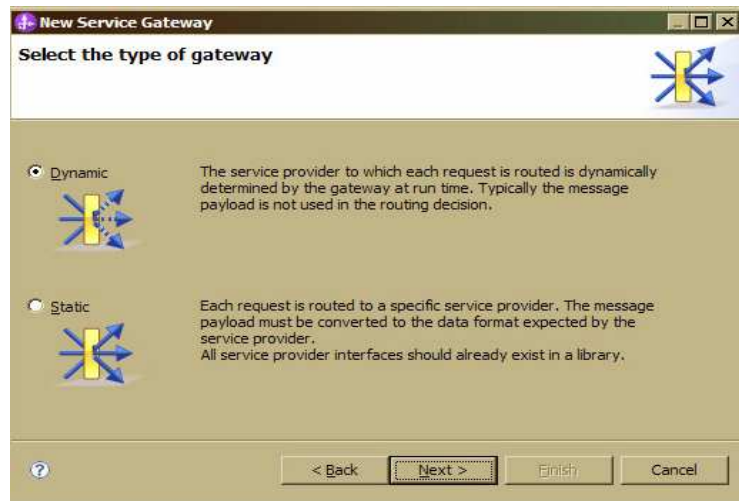
Import: CarReservation (Web Service Binding)

Click the Add button to add a JAX-WS handler. [More...](#)

Description			
Details	CarReservationHandler	Add	Select a t
Qualifiers		Delete	Name:
Binding			Class:
Policy Sets			Description:
JAX-WS Handlers		Reorder	
Propagation			

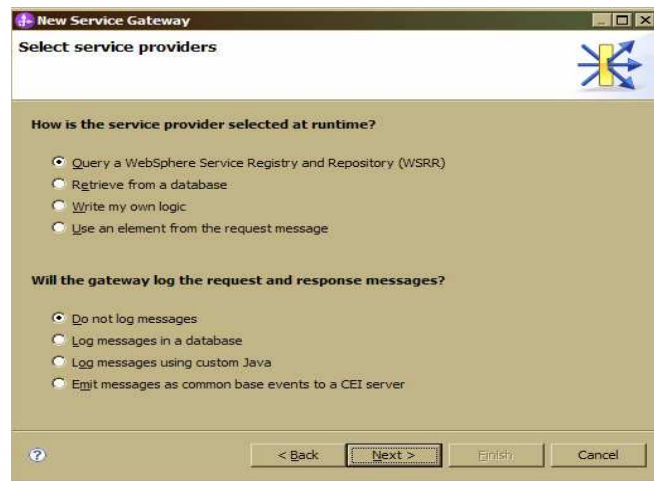
With the introduction of the service gateway as a proxy in between the client and the service provider, the clients are now redirected to the service gateway. The service gateway looks up the service registry based on the service request type provided by the client. The client can provide the service request type information in different ways such as WS addressing information, SOAP action, SOAP header, or URL information. In this example, the service request type is provided as a SOAP header. A JAX-WS handler adds the SOAP header during the Web service invocation.

Creation of dynamic service gateway (1 of 3)



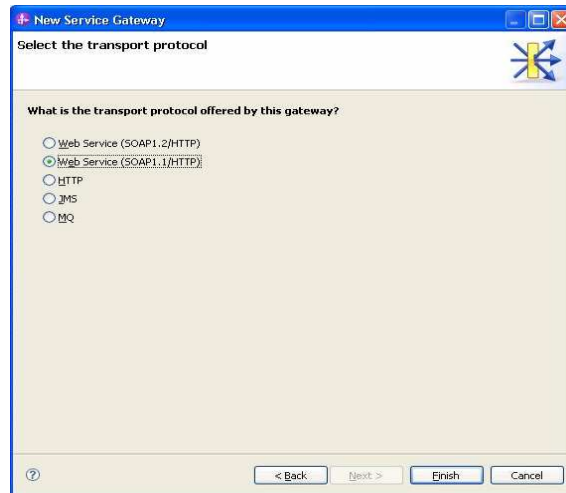
WebSphere Integration Developer allows you to create a module of type service gateway. The service gateway wizard provides the options to choose the type of service gateway.

Creation of dynamic service gateway (2 of 3)



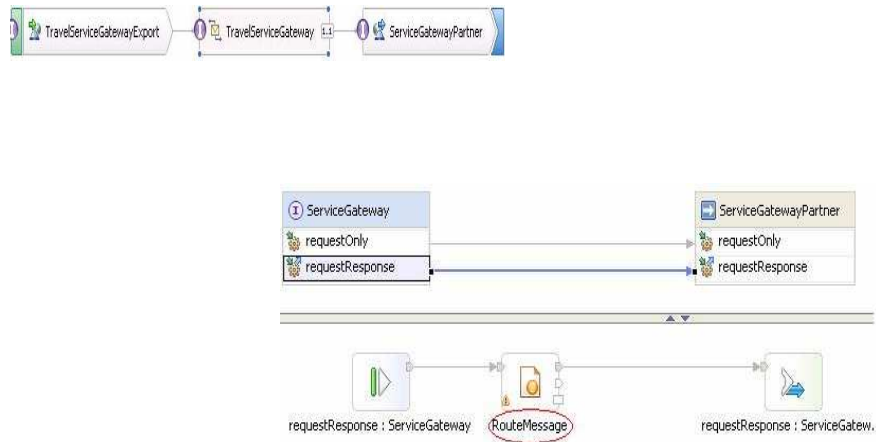
Wizard lists different options to choose the mechanism to select the service provider address.

Creation of dynamic service gateway (2 of 3)



The service gateway supports different transport protocols such as Web service binding, HTTP binding, JMS binding, and MQ binding. The example application already has a service gateway created for you.

Inside service gateway



The service gateway has a mediation component with a service partner and a Web service export. Since message logging was not selected, the mediation primitive contains an input request directly connected to the endpoint lookup primitive. The endpoint look up primitive is responsible for looking up the service registry and selecting the service provider for request message.

Endpoint lookup (RouteMessage) configuration

Endpoint Lookup : RouteMessage

Description

Terminal Name:

Details

Advanced Namespace:

Promotable Properties Registry Name: WSRRTTEST

Match Policy: Return first matching endpoint and set routing target

Advanced

Promotable Properties

User Properties:

Name	Type	Value
targetService	XPath	<code>{headers/SOAPHeader[1]}/value/targetService</code>

The request type provided by the Client is the identifier that is unique within a particular service gateway for the type of message. This service identifier can be passed into WebSphere Service Registry and Repository using the user properties section. The endpoint look up primitive looks up the service registry based on the user property. Depending on the match policy selected, one or more service provider endpoints can be selected.

WebSphere Service Registry and Repository definition

WSRR definitions

A WebSphere Service Registry and Repository (WSRR) definition enables an application or server runtime component to access a WSRR instance.

Preferences

New Delete Set as Default Test connection Clear all active caches

Select	Name	Description	Default
<input type="checkbox"/>	TESTWSRR		Yes

Total 1

Having the endpoint lookup primitive configured to use WebSphere Service Registry and Repository to look up the Web service endpoints, a definition needs to be created in the WebSphere Process Server or WebSphere Enterprise Service Bus runtime. These runtime environments provide the connectivity to the service registry through a WebSphere Service Registry and Repository definition. That definition can be created using the administrative console.

WebSphere Service Registry and Repository connection properties

The image displays two screenshots of the WebSphere Service Registry and Repository configuration interface. The left screenshot shows the 'General Properties' section with the 'Timeout of cache' field set to 0, which is circled in red. The right screenshot shows the 'General Properties' section with the 'Registry URL' field set to 'http://localhost:9089/WSRRCoreSDO/services/WSRRCoreSDOPort'.

The cache timeout controls how long results from queries that are stored in the cache are used before the query is reissued.

Loading WSDL in to WebSphere Service Registry and Repository

- WebSphere Service Registry and Repository can be accessed at:

<http://host:port/ServiceRegistry/Login.do>

The screenshot shows a dialog box titled "Path to the Document". It has two radio buttons: "Local file system" (selected) and "Remote file location". Under "Local file system", there is a "Specify path" field containing the text "C:\CarReservation_CarReservationExport1.wsdl" and a "Browse..." button. Under "Remote file location", there is a "Specify URL" field. Below these options, there is a "Document type" dropdown menu set to "WSDL", an "Enter document description:" field, and an "Enter document version:" field. An "OK" button is at the bottom left of the dialog.

Once a WebSphere Service Registry and Repository definition is created, the repository is expected to have service endpoint information. This requires you to log in to WebSphere Service Registry and Repository and load the WSDL documents. When you load a WSDL document, any XML schema upon which the WSDL depends will also be loaded and the necessary relationship between them will be established.

Add property to the service port

Ports

This is the collection of WSDL Ports present in the registry.

Preferences

Select	Name
<input checked="" type="checkbox"/>	CarReservationExport1_CarReservationHttpPort

Properties

Ports > Add Property to Object

Details of a property.

Details

General Properties

Name
targetService

Value
CarReservation

18 Service gateway in conjunction with WebSphere Service Registry and Repository © 2010 IBM Corporation

Searching the WSDL ports lists all the ports stored in the service registry. Selecting the port, and clicking the Add Property button allows you to define a new property and value.

Execution of example application

- Import the attached PI in to WebSphere Integration Developer
- Start the server and login to administrative console
- Navigate to service integration -> WebSphere Service Registry and Repository Definitions and create a new definition as shown in slide 15
- Login to WebSphere Service Registry and Repository console
host:port/ServiceRegistry/Login.do
- Load these WSDL files and their interfaces in to WebSphere Service Registry and Repository
 - CarReservation_CarReservationExport1.wsdl
 - CarReservation.wsdl
 - CreditCardChecking_CreditCardCheckingExport1.wsdl
 - CreditCardChecking.wsdl
 - FlightReservation_FlightReservationExport1.wsdl
 - FlightReservation.wsdl
 - HotelReservation_HotelReservationExport1.wsdl
 - HotelReservation.wsdl

Make sure the listed instructions are followed before you deploy the application to the WebSphere test environment server in WebSphere Integration Developer.

Execution results

Detailed Properties

Configuration: Default Module Test

Module: TravelBooking

Component: TravelBooking

Interface: TravelBooking

Operation: book

Initial request parameters:

Value editor XML editor

Name	Type	Value
request	TravelBookingRequest	✓
residence	string	✓ sfo
destination	string	✓ la
dateOfDeparture	string	✓ 10/16/2009
dateOfReturn	string	✓ 10/25/2009
creditCardNumber	string	✓ 4567435890675
creditCardCompany	string	✓ ABC
departureAirport	string	✓ sfo
departureTime	string	✓ 4PM
destinationAirport	string	✓ lay

Deploy the application and test the TravelBooking module to see the result shown above.

Summary

- Examined the Dynamic Service Gateway
 - Implementation of Dynamic service gateway
 - Configuring the endpoint look up with WebSphere Service Registry and Repository
 - Definition of WebSphere Service Registry and Repository in the WebSphere Process Server runtime
 - Loading the WSDL documents and adding properties to the service ports
 - Sample scenario with an example application

In summary, this presentation introduced you to an overview of the service gateway and the differences between the static and dynamic service gateways. A dynamic service gateway, in conjunction with WebSphere Service Registry and Repository was examined. You learned about how to create a service gateway and configure the endpoint look up primitive with WebSphere Service Registry and Repository. You also learned about how to create WebSphere Service Registry and Repository definitions in WebSphere Process Server or WebSphere Enterprise Service Bus runtime using the administrative console. You were introduced to a sample application that used a service gateway to dynamically select the service endpoints at runtime. You finally learned how to load the WSDL documents to WebSphere Service Registry and Repository and add properties to the service ports.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_Service_Gateway_with_WSRR.ppt

This module is also available in PDF format at: [./Service_Gateway_with_WSRR.pdf](http://Service_Gateway_with_WSRR.pdf)

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the Web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2010. All rights reserved.