BEA AquaLogic®
Service Bus

BEA AquaLogic Service Bus Interoperability Solutions for WSRP

Version: 3.0
Revised: February 2008

Contents

Introduction

Web Services for Remote Portlets (WSRP) is a mechanism used to generate markup fragments on a remote system for display in a local portal application.This chapter describes how AquaLogic Service Bus provides Service Level Agreement (SLA) monitoring in applications that use WSRP.

This section discusses the following topics:

- WSRP Producers and Consumers
- WSRP Architecture
- WSRP Design Concepts

WSRP Producers and Consumers

WSRP involves two integral components:

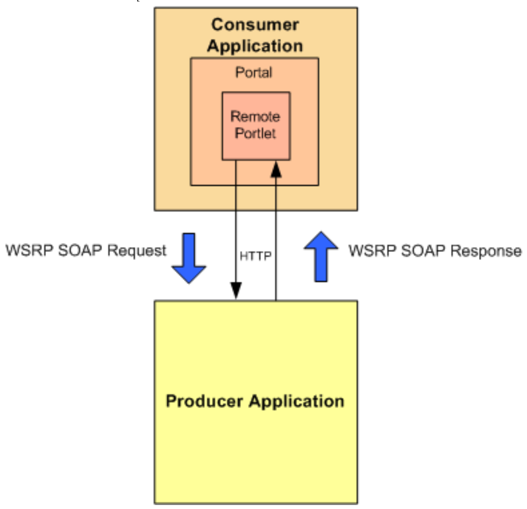
- *WSRP producer* - (referred to as *producer* in this document) is a remote application, implements standards-based Web Services using the SOAP specification over HTTP. You can create a producer using WebLogic Portal or third-party implementations of WSRP.
- *WSRP consumer* (referred to as a *consumer* in this document) is a portal application. Typically, the consumer application references WSDL of the producer when the portal is designed, and the consumer directly accesses the producer.

WSRP Architecture

This section describes the architecture of WSRP and shows how to enhance the architecture by adding AquaLogic Service Bus.

Figure 1-1 shows the WSRP SOAP request and response flow between a producer application and a consumer application.

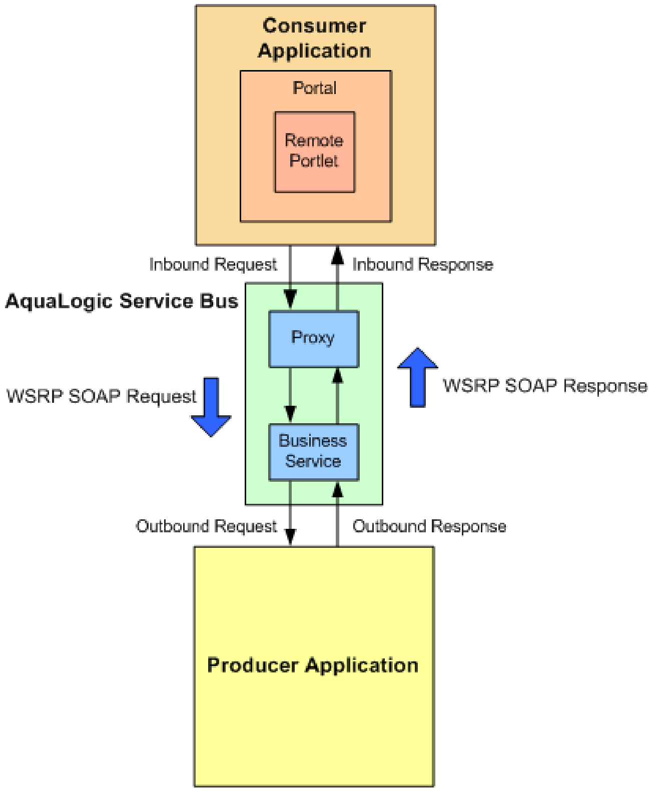
Figure 1-1 Basic Request/Response Flow Between Producer and Consumer Applications



Enhanced Architecture with AquaLogic Service Bus

Figure 1-2 shows how to use AquaLogic Service Bus as an intermediary between the producer and the consumer to provide Service Level Agreement (SLA) monitoring. You can use AquaLogic Service Bus in this.

Figure 1-2 Enhanced WSRP Request / Response Flow via AquaLogic Service Bus



The WSRP SOAP request/response flow occurs in the following sequence:

1. **Inbound Request:** The consumer calls the proxy service in AquaLogic Service Bus.
2. **Outbound Request:** The proxy service routes the request, which is a message containing the SOAP body and transport headers, to the producer
3. **Outbound Response:** The producer sends a response to AquaLogic Service Bus.
4. **Inbound Response:** The proxy service sends a response to the consumer. The response is a message that contains the SOAP body and transport headers.

The remainder of this section describes how to configure AquaLogic Service Bus to send requests for WSRP services through proxy services. It describes services that a producer provides, along with other attributes of WSRP that must be used to configure AquaLogic Service Bus. This section also discusses how to monitor producers with increasing degrees of detail. Finally, it discusses load balancing and failover with WSRP.

WSRP Design Concepts

This section describes the following WSRP design concepts:

- WSRP WSDLs
- WSRP Messages

WSRP WSDLs

Table 1-1 describes various kinds of services offered by WSDLs. And WSDLs are referred to as ‘Producers’.

Table 1-1 Producer Services

Service	Description
Service Description	*Required service.* Describes the producer and the portlets that the producer makes available to consumers.
Markup	*Required service.* Manages user interaction with a remote portlet and returns the HTML markup used to render the portlet.
Registration	*Optional service.* Required for complex producers. Allows consumers to register themselves with the producer.
Management	*Optional service.* Provided by complex producers for managing portlet customization and portlet preferences.
Markup Extension	Provided by BEA Portal producers and replaces the Markup service. Markup Extension allows more efficient message handling by using multipart MIME messages for transmitting HTML markup content.

Each producer implements a minimum of two services, such as Service Description and Markup A simple producer offers just these two services. A complex producer, however, provides two additional services, such as Registration and Management. In addition, WebLogic Portal producers implement an extension service, such as Markup Extension that replaces the standard Markup service.

These services are described using a standard WSDL format. The producer supplies a single URL for retrieving its WSDL, which describes all the services that are provided by that producer. The end points for each service indicate whether the consumer should use transport-level security (HTTPs) or abstain from communication with the producer.

WSRP Messages

WSRP uses SOAP over HTTP for all messages exchanged between producers and consumers. In addition to using the standard message formats in the SOAP body, WSRP requires that consumers set at least a `SOAPAction` header, the cookie headers, and the usual HTTP headers, such as `Content-Type`. Producers return a session cookie, and any application-specific cookies, in the HTTP transport header of the response. The consumer must return the session cookie in subsequent request messages.

Configuring AquaLogic Service Bus for WSRP

The AquaLogic Service Bus Console, which is described in *Using the AquaLogic Service Bus Console*, is used to configure AquaLogic Service Bus. For more information about creating WSRP-enabled portals by using WebLogic Portal, see *Federated Portals Guide*.

Configuring AquaLogic Service Bus for WSRP involves the following tasks:

- Implementing a service that consumers can invoke to obtain an appropriate WSDL for a particular producer.
- Implementing the details of conveying a consumer's request to the producer and returning the response to the consumer.

This chapter describes the following tasks:

- Getting the Producer WSDL
- Routing Messages Between Consumer and Producer
- Monitoring WSRP Applications
- Load Balancing and Failover

Getting the Producer WSDL

As a common practice, consumers directly contact a producer to obtain its WSDL. However, if AquaLogic Service Bus is used as a proxy service, then all access to the producer occurs via AquaLogic Service Bus. Therefore, a proxy service is implemented for consumers. The proxy service calls the producer's real URL to obtain the producer WSDL. The proxy service transforms the results as follows:

- Rewrite the endpoint address for the producer to refer to the Service Bus IP address and port
- Change the endpoint URI to refer to the AquaLogic Service Bus proxy service that reflects the required monitoring granularity as described in Monitoring WSRP Applications
- Change the endpoint protocol and port if transport security is used between the consumer and the AquaLogic Service Bus proxy service

The developer who creates a producer can specify whether the producer requires SSL (`"secure=true"`). In addition, the AquaLogic Service Bus administrator can change the security requirement to the consumer via AquaLogic Service Bus configuration. For example, if a producer does not require SSL, the AquaLogic Service Bus administrator can require consumers to use SSL by doing the following:

- Changing the WSDL to specify HTTP(s)
- Configuring the proxy services for WSRP to use HTTP(s)

When configured in this way, AquaLogic Service Bus automatically bridges the secure messages from the consumer to the non-secure messages used by the producer.

Routing Messages Between Consumer and Producer

After retrieving a copy of the WSDL, the consumer uses the WSDL definitions to formulate service requests and sends them to the producer via AquaLogic Service Bus. The WSRP request/response process involves the following steps:

1. The consumer sends a message to the AquaLogic Service Bus proxy service corresponding to the producer service.
2. The proxy service executes a simple message flow that routes the message (unchanged) to the actual producer service.
3. The producer formulates and sends a response to the consumer via AquaLogic Service Bus.
4. The consumer receives the response (unchanged) from the producer.

WSRP web services expose portlets, and those can rely on HTTP cookies and sessions. You must configure WLSB to propagate HTTP transport headers, such as `SOAPAction` and cookies. However, by default, AquaLogic Service Bus does not pass transport headers from the proxy service to the business service because the proxy service may not use the same transport as the business service. Therefore, you must configure the message flow to copy the request headers from the inbound request to the outbound request. Similarly, you must copy the response headers from the business service back to the proxy service's response to the consumer.

Although it is possible to copy all transport headers between the proxy service and the business service, it is necessary to be more selective to avoid errors. You must copy the Set-Cookie and Cookie headers. The final message must own some headers, such as `Content-Length` because AquaLogic Service Bus is the entity that assembles the final message to send. For example, if the message flow copies the `Content-Length` header from the proxy service to the business service, it can result in an error because the length of the message can change during processing. Therefore, ALSB must own this header.

Monitoring WSRP Applications

Monitoring a WSRP application tracks the usage of a producer's individual services and operations. The message flow for WSRP services introduces very little overhead, and the mapping between proxy services and producers, and between business services and producers, is simple to configure. Therefore, to satisfy SLA requirements, it is sufficient to monitor only the proxy services.

For more information about monitoring WSRP Applications, see Monitoring ALSB at Run Time in *Operations Guide*.

Load Balancing and Failover

AquaLogic Service Bus allows business services to define multiple endpoints that provide the same web service. When multiple endpoints are defined, AquaLogic Service Bus can automatically distribute load balance requests across endpoints, and it can automatically failover requests when an endpoint is inaccessible. However, WSRP imposes some limitations on the use of these features.

Portlets are a means of exposing a user interface to an application. Therefore, portlets typically have session data associated with them. To preserve session data, requests to the portlet must be directed to the same server or cluster that serviced the original request. This requirement makes load balancing via AquaLogic Service Bus inappropriate. Multiple endpoints in a business service usually target different servers or clusters. There is no way to preserve the session because there is no communication among servers that are in separate clusters. Therefore, if multiple endpoints are defined for a WSRP business service, then you must set the load-balancing algorithm to `"none"`.

You can use multiple endpoints to provide redundancy in certain circumstances if the event that one of the endpoints is not available. The WSRP service is still available on a secondary endpoint. However, any session data that existed at the time the first endpoint failed will not be available on other endpoints.

This failover configuration is an option only for simple producers (see WSRP WSDLs), not for complex produces. Complex producers require that their consumers register with the producer before sending service requests. The producer returns a registration handle that the consumer must include with each request to that producer. If a business service defines multiple endpoints, each endpoint provides and requires its own registration handle.

However, AquaLogic Service Bus is stateless across requests—it does not maintain a mapping of the correct handle to send to a particular endpoint. In fact, it sends the registration request to a single endpoint, so the consumer is registered with only one producer. If that producer is not available, then AquaLogic Service Bus routes a service request to another endpoint defined for that business service. However, the consumer is not registered with that new producer, and the request fails with an `"InvalidRegistration"` fault.

The management of registration handles requires an application outside of AquaLogic Service Bus to maintain this state data. Therefore, the registration requirement avoids defining multiple endpoints for complex producers. As simple producers do not support the registration service, a failover configuration that defines multiple endpoints in the business service is possible although session data is lost on failover.

WSRP Interoperability Example

The examples described in this section support **AquaLogic Service Bus** versions 2.0, 2.1, 2.5, and 3.0 (see **Example Prerequisites**).

This section describes a WSRP 2.0 interoperability example. For an example of WSRP 1.0 interoperability, see WSRP Interoperability Example in *BEA AquaLogic Service Bus 2.6 Documentation.*

This section discusses the following topics:

- Example Prerequisites
- Example Projects and Folders
- Monitoring Example

Example Prerequisites

The WSRP interoperability example assumes the following components and configuration:

- WebLogic Platform 10.2
- WebLogic Portal 10.0
- AquaLogic Service Bus 3.0
- Sample Platform domain configured at `platform:7001`
- AquaLogic Service Bus domain configured at `alsb:7001`

- Sample Portal application consumer
- Sample producer

For an AquaLogic Service Bus configuration that supports the configuration defined in this example, see the AquaLogic Service Bus/WSRP code sample, available from the AquaLogic Service Bus code samples page on BEA dev2dev:

- For AquaLogic Service Bus 3.0, see:
  `https://codesamples.projects.dev2dev.bea.com/servlets/Scarab/remcurreport/true/template/ViewIssue.vm/id/S403/nbrresults/13`
- For AquaLogic Service Bus 2.5, see:
  `https://codesamples.projects.dev2dev.bea.com/servlets/Scarab/remcurreport/true/template/ViewIssue.vm/id/S267/nbrresults/13`
- For AquaLogic Service Bus 2.0 and 2.1, see:
  `https://codesamples.projects.dev2dev.bea.com/servlets/Scarab/remcurreport/true/template/ViewIssue.vm/id/S175/nbrresults/13`

Example Projects and Folders

This example describes the configurations that are relevant to the ALSB 3.0 dev2dev code sample.

The structure of the sample is divided into two projects—one containing common resources, and the other containing resources for the sample producer.

Table 3-1 Projects in the WSRP Interoperability Examples

Folder	Description
`wsrp`	Contains common resources that are not specific to any producer.
`operationExample`	Full example supports the most fine-grained monitoring. The folder contains resources specified by the producer. See Monitoring Example.

Monitoring Example

The monitoring configuration example (in the `operationExample folder`) involves configuring AquaLogic Service Bus to monitor all services and operations of a producer.

The monitoring configuration uses both business services and proxy services that are based on the WSDLs defined by the WSRP standard. The section discusses the following topics:

- Step 1: Define WSDL Resources
- Step 2: Create Business Services
- Step 3: Create Proxy Services
- Alternative Methods to Create Proxy Services
- Step 4: Retrieve the WSDL from the Producer
- Step 5: Verify the Configuration

Step 1: Define WSDL Resources

Import all the WSRP WSDL definition files, along with the XML schema files on which the definitions depend. All the files are available as part of the sample code associated with this example, but the standard resource locations are listed in Table 3-2.

Table 3-2 WSDL Resource Definitions

Resource Name	Type	Location
`xml-2.0`	`XML Schema`	`http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL/xml.xsd`
`wsrp-2.0-types`	`XML Schema`	`http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL/wsrp-2.0-types.xsd`
`wsrp-2.0-interfaces`	`WSDL`	`http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL/wsrp-2.0-interfaces.wsdl`

Table 3-2 WSDL Resource Definitions (Continued)

Resource Name	Type	Location
wsrp-2.0-bindings	WSDL	http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL/wsrp-2.0-bindings.wsdl
wsrp-2.0-wsdl	WSDL	http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL

Producers generated by BEA Portal extend the standard WSDLs by defining an additional port that allows messages to be sent using MIME attachments. You must define these extension resources if the producer WSDL references them. In this example, an optional task is to create a resource for the WSDL used by the producer. After creating these WSDL and XML Schema resources, edit the references in each resource to resolve the dependencies on other resources.

Step 2: Create Business Services

This monitoring example uses the WSDL bindings for each port type implemented by the producer. You must create a separate business service resource for each business service because a business service can be associated with only one WSDL port or binding. A simple producer implements only the required Markup and Service Description interfaces, while a complex producer also implements the Management and Registration interfaces. The services are created identically except for the service name and types, see Table 3-3.

Table 3-3 Business Service Configuration

Service Name	Service Type
base	WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port="WSRPBaseService"
desc	WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPServiceDescriptionService"

Table 3-3 Business Service Configuration (Continued)

Service Name	Service Type
`mgmt`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPPortletManagementService`
`reg`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPRegistrationService"`

For each service, the required attributes are listed in Table 3-4.

Table 3-4 Service Attributes for Business Services

Name	Value	Comments
Protocol	`HTTP`	Or HTTP(s) if the producer was created with secure= `true`.
Load Balancing Algorithm	`none`	Must be `none`, or session data will be lost across requests if multiple end points are defined.
Endpoint URI	• Service Description: `http://host*:port+/producer/producer/wsrp-2.0/serviceDescription`	Multiple endpoints must be defined for WSRP producers.
	• Markup: `http://host*:port+/producer/producer/wsrp-2.0.0/markup`	
	• Registration: `http://host*:port+/producer/producer/wsrp-2.0/registration`	
	• Portlet Management: `http://host*:port+/producer/producer/wsrp-2.0/portletManagement`	

Step 3: Create Proxy Services

Proxy services in this monitoring example are configured as follows:

- The proxy services must be based on the same WSDL because the business services are based on a WSDL.
- One proxy service is created for each business service, but each proxy service must have a different URI.
- The configuration must specify which operation is being invoked.

To create a proxy service:

1. Create the proxy service for the base WSRP service.

   As in the earlier example, create the proxy service using the existing `operationExample/base` business service as the model. This model creates the proxy servicey based on the same WSDL binding as the business service, and it creates a message flow with an unconditional route action to the business service. For the endpoint URI, you can use any URI, such as the producer name with the port type abbreviation appended to it (for example, `/operationExampleBase`).

2. Edit the message flow to add the same transformations required to copy the request transport headers and response transport headers between the consumer and producer.
   WSRP relies on data conveyed in the transport headers to function properly. In particular, producers return session cookies to consumers in the response headers if they expect consumers to supply session cookies in subsequent requests. Similarly, producers expect consumers to provide the requested operation in the `SOAPAction` request header.

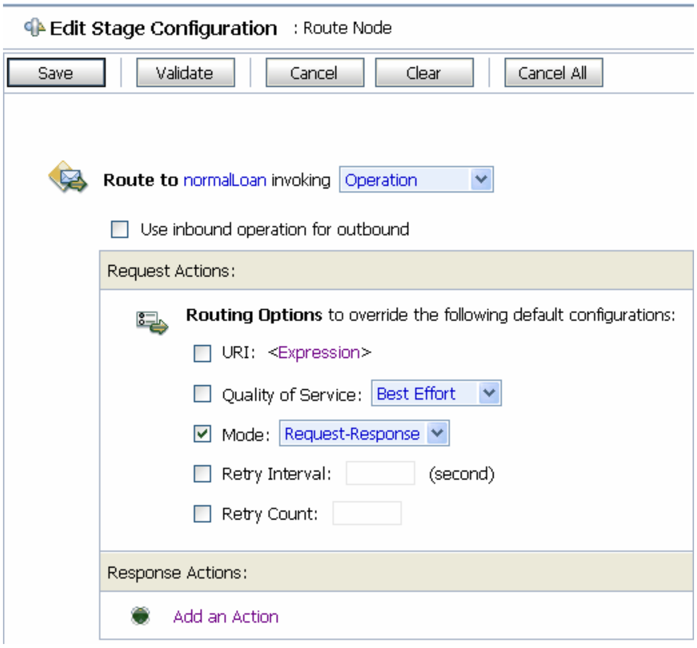
   By default, AquaLogic Service Bus does not copy transport headers from the inbound request to the outbound request, or from the outbound response to the inbound response. The message flow must propagate the required headers both in and out of the business service.

   Note: To retrieve all the headers from the transport, select **Yes** in the **Get All Headers** field of the **Transport Configuration** page of the ALSB proxy. For more information, see Transport Configuration Page in the *Using the AquaLogic Service Bus Console*.

   Add Transport headers to the **Request & Response** actions of the **Route Node** in the Message Flow and enable the **Pass all headers through pipeline option**. For more information, see Adding Transport Header Actions in *Using the AquaLogic Service Bus Console*. ALSB automatically ensures that content-length is not copied.

3. Alternatively, while configuring the routing to a business service using the AquaLogic Service Bus console, select the **Use inbound operation for outbound** check box when you are editing a route node to avoid low level Xquery manipulation, as in Figure 3-1.

Figure 3-1 Passing an Operation from Inbound to Outbound



Route to normalLoan invoking Operation

☐ Use inbound operation for outbound

Request Actions:

Routing Options to override the following default configurations:

☐ URI: <Expression>

☐ Quality of Service: Best Effort

☑ Mode: Request-Response

☐ Retry Interval: (second)

☐ Retry Count:

Response Actions:

With this transformation, the operation for the business service is dynamically set to the same value as was specified for the proxy service. AquaLogic Service Bus counts and monitors all operations of the service.

Alternative Methods to Create Proxy Services

The proxy services for the other business services can be created by repeating these steps although you can use a shortcut to avoid recreating all of the transformations manually.

For example, to create the proxy service for the Service Description service:

1. Create a new proxy service using the existing `operationExample/base` proxy service that is created as the model. Following this example, use `/operationExampleDesc` for the endpoint URI.
2. On the Summary Page, click the **Edit** link for General Configuration. The WSDL binding is created using the Base port, so modify the binding to refer to the `WSRPServiceDescriptionService` port.

3. Edit the message flow. The route action refers to the base business service. Modify the route action to the `desc` service.

Note: Use the Transport Header action to minimize low level Xquery manipulation, and simplify the configuration of a proxy service. See Transport Headers section in the AquaLogic Service Bus *Console Online Help* for details.

For each service, the proxy service configurations are listed in Table 3-5.

Table 3-5 Proxy Service Configuration

Service Name	Service Type
`proxyBase`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port="WSRPBaseService"`
`proxyDesc`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPServiceDescriptionService"`
`proxyMgmt`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPPortletManagementService"`
`proxyReg`	`WSDL port: operationExample-2.0/wsrp-2.0-wsdl, port=" WSRPRegistrationService"`

For each proxy service, the required attributes are listed in Table 3-7.

Table 3-6 Required Attributes

Name	Protocol
Protocol	HTTP
Get All Headers	Yes
Endpoint URI	For WebLogic Platform 10.0/9.2 the URLs are as follows: • proxyBase: `/operationExampleBase-2.0` • proxyDesc: `/operationExampleDesc-2.0` • proxyReg: `/operationExampleReg-2.0` • proxyMgmt: `/operationExampleMgmt-2.0`

Step 4: Retrieve the WSDL from the Producer

Create a service that retrieves the WSDL specific to WSRP 2.0 from the producer and transform it to hide the actual producer endpoints. In this example, the proxies for each producer have a different URI. In addition, the section describes how to create the resources to retrieve the producer WSDL.

Step 4.1: Create the Business Service to Retrieve the WSDL Specific to WSRP 2.0

Create a business service to obtain the WSDL from the producer. This resource is specific to the producer, so you must create the resource in the operationExample project. Table 3-7 describes the properties of the business service.

Table 3-7 Business Service Configuration Properties

Name	Value	Comments
Service Name	`wsdlSvc 2.0`	Any name is allowed.
Service Type	Any XML Service	Consumers usually retrieve the WSDL from the producer using an HTTP GET request. Only XML services support GET.
Protocol	HTTP	HTTP
Load Balancing Algorithm	None	None preferred.
Endpoint URI	`http://platform:7001/producer/producer/wsrp-2.0/markup?WSDL`	Although you can specify multiple endpoints for retrieving the WSDL, it has no additional use or benefit.
HTTP Request Method	GET	

Step 4.2: Create an XQuery Expression to Construct URLs

You must transform all endpoint addresses in the producer's WSDL to reflect the AquaLogic Service Bus server address and the proxy service URI values. You must create an XQuery expression to simplify the construction of the endpoint locations because each producer WSDL can have four or more ports defined. The XQuery expression accepts the following three string variables as input and concatenates them together to form a SOAP address element:

- **base URL** for the AquaLogic Service Bus server
- **name** to identify the producer
- **extension** used to differentiate ports for a producer

Table 3-8 provides the query definition in the `wsrp` project.

Table 3-8 XQuery Definition in the wsrp Project

Name	Value
Resource Name	`wsrp/addr`

XQuery	`declare variable $baseURL external;`
	`declare variable $name external;`
	`declare variable $svc external;`
	`declare namespace soap="http://schemas.xmlsoap.org/wsdl/soap/";`
	`<soap:address location="{concat($baseURL, $name, $svc)}"/>`

Step 4.3: Create a No-Op Proxy Service

Create a service that does nothing. To create this service, define a new proxy service in the `wsrp` project folder with the resource name `nullSvc`. Accept all of the defaults for this service. Configuring this proxy service creates a message flow for the service of an echo node.

Step 4.4: Create a Common Proxy Service to Retrieve the WSDL Specific to WSRP 2.0

Create a proxy service used by consumers to get WSDLs from producers. This proxy service is appropriate for any producer configuration modeled on this sample. The example described in this section is only a suggestion–a different approach is necessary based on the specific requirements of a given implementation. You must create this proxy service in the `wsrp` project folder because the proxy service is not specific to a single producer.

- The approach used in this step requires the administrator to assign each producer a name that is included in part of the URL to retrieve the WSDL.

- The message flow for the proxy service extracts the name from the URL, uses it to locate the business service specific to that producer, obtain the WSDL, and then transforms the WSDL to rewrite the endpoints to AquaLogic Service Bus.
- The proxy service endpoint URI is configured as `/getWSDL`, and the URL that consumers use to obtain a WSDL is as follows:

  ```
  http://alsb:7001/getWSDL/<producerName>
  ```

  where `<producerName>` is the name assigned to the producer by the administrator. In this example, the producer is `operationExample`.

Table 3-9 describes the configuration properties for the proxy service `getWSDL2.0` :

Table 3-9 Proxy Service Configuration Properties

Property Name	Value	Comments
Service Name	getWSDL2.0	Any name is allowed.
Service Type	Any XML Service	
Protocol	HTTP	
Endpoint URI	`/getWSDL2.0`	

The message flow for this proxy service consists of a pipeline pair and a route node. The request side of the pipeline pair consists of a single stage whose job is to extract the producer name from the URL and assign it to a context variable. The action is:

```
Assign $inbound/ctx:transport/ctx:request/http:relative-URI to variable
producerName
```

The response side of the message flow is a stage where all the transformations are performed. Before executing the Replace Actions to transform the WSDL, assign the base URL of the AquaLogic Service Bus server to a context variable to avoid specifying it on every transformation:

```
Assign "http://alsb:7001/" to variable nonSecureBaseURL
```

Edit the stage of the Response Pipeline to modify each Replace Action to make the transformation match the Endpoint URI given to the proxies created earlier. In this example, the proxies were created using the producer name with an abbreviated service type appended to it. The `addr` XQuery resource created earlier accepts an extension argument to construct the URI location. Simply change that argument to the proper value, as listed in Table 3-10.

Table 3-10 Extension Settings to Construct the URI Location

If `@binding` is	svc arg of `addr` is
`WSRP_v2_Markup_Binding_SOAP`	`"Base"`
`WSRP_v2_ServiceDescription_Binding_SOAP`	`"Desc"`
`WSRP_v2_PortletManagement_Binding_SOAP`	`"Mgmt"`
`WSRP_v2_Registration_Binding_SOAP`	`"Reg"`
`WLP_WSRP_v2_Markup_Ext_Binding_SOAP`	`"Ext"`

You must map `name:` to `$producerName` and `BaseURL` to `$nonSecureBaseURL` similar to the `svg arg` mapping in the use table: table `num_xref`, Extension Settings to Construct the URI Location.

The five Replace Actions are defined, as in the following code listing. The value of name is replaced with the binding names from the table.

```
Replace
./*[local-name()="definitions"]/*[local-name()="service"]/*[local-name()="
port"][ends-with(attribute::binding,"name")]/*[local-name()="address"

Replace entire node

name

WSRP_v2_Markup_Binding_SOAP

WSRP_v2_ServiceDescription_Binding_SOAP

WSRP_v2_PortletManagement_Binding_SOAP

WSRP_v2_Registration_Binding_SOAP
```

For the first Replace Action, you must add the User Namespace definitions listed in Table 3-11:

Table 3-11 User Namespace Definitions on Replace Action

Prefix	Namespace
`wsdl`	`http://schemas.xmlsoap.org/wsdl/`
`soap`	`http://schemas.xmlsoap.org/wsdl/soap/`

The route node of this message flow consists of a routing table that selects the case based on `$producerName`. For each known producer, add cases so that each case routes to the correct business service to retrieve the WSDL if the name matches. This example uses the following directive:

```
= "operationExample" Route to wsdlSvc
```

1. Add a Default Case that routes to the no-op service to handle cases in which an unknown producer name is specified:
   `Default Route to nullSvc`
2. In this example, return an HTTP 404 status code by adding these response actions to the default case:

   ```
   Insert <http:http-response-code>404</http:http-response-code> as last
   child of ./ctx:transport/ctx:response in variable inbound

   Reply With Failure
   ```
3. Edit the Routing Table in the route node to make the cases correspond to the producers known to the system.

Step 4.5: Define the Message Flow of getWSDL2.0 Proxy Service to Modify WSDL to Use ALSB URLs

The message flow for `getWSDL2.0` proxy service consists of a pipeline pair and a route node. To define the message flow for this proxy service, do the following:

1. Create a Pipeline Pair
2. Edit the request side of pipeline pair

The request side of the pipeline pair consists of a single stage whose job is to extract the producer name from the URL and assign it to a context variable. To achieve this, add the Assign action as follows:

Assign `$inbound/ctx:transport/ctx:request/http:relative`-URI to variable producerName

3. Edit the response side of pipeline pair

The response side of the message flow is a stage where all the transformations are performed.

- Create an Assign action to assign the base URL of the ALSB server to context variable to avoid specifying it on every transformation:

Assign `http://alsb:7001/` to variable nonSecureBaseURL

- Create Replace actions to change the URLs in the response WSDL to make them point to the corresponding proxy services created earlier. Add the Replace action as follows:
  - Add Replace action.
  - In the Xpath, specify `./*[local-name()="definitions"]/*[local-name()="service"]/*[local-name()="port"][ends-with(attribute::binding,"WSRP_v2_Markup_Binding_SOAP")]/*[local-name()="address"][starts-with(attribute::location,"http:")]`
  - In text box, specify variable body.
  - In the Expression, Select the Xquery resource addr (which was added earlier as a part of wsrp 1.0 configuration) and set svc to `Base-2.0`, baseURL to $nonSecureBaseURL, and name to $producerName.
  - Select the option **Replace entire node**.
  - Create the remaining Replace actions, as described in the following.

Table 3-12

If attribute binding in Xpath is	svc argument of addr Xquery will be
`WSRP_v2_Markup_Binding_SOAP`	`"Base-2.0"`
`WSRP_v2_ServiceDescription_Binding_SOAP`	`"Desc-2.0"`
`WSRP_v2_PortletManagement_Binding_SOAP`	`"Mgmt-2.0"`
`WSRP_v2_Registration_Binding_SOAP`	`"Reg-2.0"`

4. Add the route node to the pipeline pair. The route node of this message flow consists of a routing table that selects the case based on `$producerName`. For each known producer, add cases so that each case routes to the correct business service to retrieve the WSDL if the name matches. This example uses the following directive:

```
= "operationExample" Route to wsdlSvc-2.0
```

   a. Add a **Default Case** that routes to the no-op service to handle cases in which an unknown producer name is specified:

```
Default Route to nullSvc
```

a. In this example, return an HTTP 404 status code by adding these response actions to the default case:

b. Insert `<http:http-response-code>404</http:http-response-code>` as last child of **./ctx:transport/ctx:response** in variable inbound

   Reply With Failure

c. Edit the **Routing Table** in the route node to make the cases correspond to the producers known to the system.

Step 4.6: Change the Message Flow of Proxy Service getWSDL to Enable WSRP 2.0 WSDL Retrieval Using Proxy Service getWSDL2.0

In order to consume the producer, the consumer must register the producer using its WSDL. By default, the WSDL uses WSRP version 1.0. To make consumer use WSRP version 2.0, producer WSDL with WSRP version 2.0 must be made available to the consumer. Generally, the default producer WSDL (one which uses WSRP 1.0) contains a link to the WSDL with WSRP version 2.0. This link enables consumer to use WSDL of WSRP version 2.0.

The getWSDL proxy service provides the WSDL of the producer with WSRP 1.0 version. This WSDL contains an URL of the WSDL of the same producer with version WSRP 2.0. This is the direct URL of the producer's WSDL. Instead of using the direct URL, you must the access the WSDL through ALSB using proxy service `getWSDL2.0`. This can be achieved in following way.

1. Create an Xquery resource to construct the URL of WSDL, which uses WSRP 2.0

   Resource Name: `import`

   Xquery:

```
declare variable $baseURL external;

declare variable $name external;

declare variable $svc external;

<import location="{concat($baseURL, $svc, $name )}"
namespace="urn:oasis:names:tc:wsrp:v2:wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/" />
```

2. Add the Replace action in the response side of message flow of `getWSDL proxy service`. This replace action replaces the WSDL URL in the response to the `getWSDL2.0` proxy service endpoint URI. This can be done as follows:

   – Add Replace action.

- In Xpath, specify `./*[local-name()="definitions"]/*[local-name()="import"][ends-with(attribute::location,"/producer/wsrp-2.0/markup?WSDL")]`
- In the textbox, specify variable body.
- In **Expression**, select the Xquery resource import and set svc to `getWSDL2.0`, baseURL to `$nonSecureBaseURL`, and name to `$producerName`.
- Select the option **Replace entire node**.

Step 5: Verify the Configuration

After completing the configuration, verify it as follows:

1. Retrieve the WSDL from a regular browser window by entering the following URL:

   `http://alsb:7001/getWSDL/operationExample`

2. Verify that all of the end point WSRP end point URLs (except for the BEA extension service) have been changed to correctly refer to the proxy service values on the AquaLogic Service Bus server.

3. Create a remote portlet in a Portal consumer application, specifying this URL as the address of the WSDL for the producer.

   Use either Workspace Studio or Portal Administration Tool to create the remote portlet. Except for entering a different URL to retrieve the WSDL, the steps to create this portlet are the same as those used to create the portlet that is not proxied by AquaLogic Service Bus.

4. After the consumer portal is complete, run the application.

5. Enable monitoring on the AquaLogic Service Bus components that you have chosen.

6. Use the AquaLogic Service Bus Console to drill down to see message counts and performance statistics on all WSRP services and operations handled by the producer.