Legal Notices

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions, Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

.

Introduction

The Outbound Services Connector (OSC) leverages the Oracle Insurance Policy Administration's extensibility framework to provide a generic, reusable approach to outbound data delivery.

The extension provides mechanisms to specify delivery targets and message templates. The latter allows users to map between the data at a given extension point and the downstream interface. Further, it avoids the need to develop individual extensions for each downstream interface, while maintaining application performance and throughput.

Before using the OSC, several installation steps must be completed.

1. Install the extension
2. Update the configuration files
3. Configure templates

After the OSC is installed, there are three additional steps that must occur to prepare for a call to the OSC.

1. Configure a service endpoint in the service registry.
2. Create a message template.
3. Configure OIPA to call the OSC extension and pass in data.

This installation document will cover all installation steps, as well as the steps involved in configuring a call to the OSC.

Customer Support

If you have any questions about the installation or use of our products, please visit the My Oracle Support web-site: https://support.oracle.com, or call (800) 223-1711.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Prerequisites

The OSC requires that the Oracle Insurance Policy Administration (OIPA) application and database be installed.

Application Servers

The Outbound Services Connector relies heavily on tools and libraries provided by the Java environment. In general, any J2EE 5+ application server with JVM 1.6 or later should provide adequate support.

- Oracle WebLogic 11g R1+ has been tested and is supported at all patch levels.
- Please note that support for JBoss is not included.

Database Storage

Version 0.3 introduced the ability to store templates in a database. Anything supported by JPA will work. For practical reasons, only databases certified with OIPA will be supported by the OSC.

JMS Providers

Most JMS providers should work with the OSC. The following configurations have been tested. If there is a client-specific request, it can most likely be accommodated.

- WebSphere MQ Version 7.0
- WebLogic Built-in Version 11gR1+

Installation

There are several steps required to install and configure the OSC and its resources.

Locating Configuration Files

There are two core OSC files that need to be placed locally on the machines hosting the OIPA application.

- osc.properties
- service-registry.xml.

Values and settings for these files will be referenced throughout the documentation. Complete references for both can be found as appendices to this document: OSC Property Values and Service Registry Tags.

OSC Properties

The osc.properties file contains settings that control how OSC will execute. This file is generally configured at the beginning of a project, and will likely be updated rarely.

This file needs to be located on the JVM classpath. This is typically achieved by adding it to the OIPA properties directory, or by creating an OSC-specific directory and adding it to the classpath.

Once this file is configured, it can generally be deployed to all environments without modification.

When configuring for WebSphere, comment out the Wallet section in osc.properties. A sample configuration of osc.properties is below:

- Service Description fileservices.file=/opt/oracle/osc/osc/service-registry.xml# File or db loading for templatestemplate.source=file# For File based loading, set the following to the templates directory, ignored otherwisefreemarker.templateDir=/opt/oracle/osc/osc/templates/

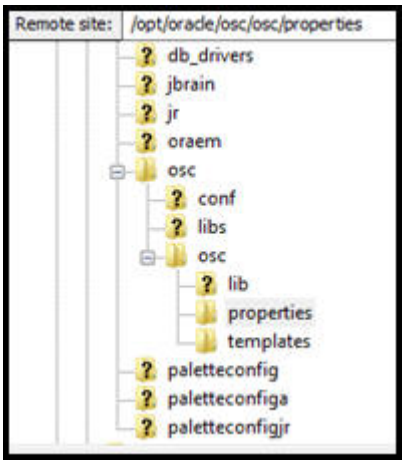


Figure 1: File structure on the server

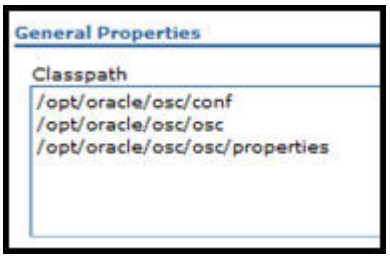


Figure 2: Classpath on the server

Service Registry

The service-registry.xml file contains a list of service definitions used in OSC. This file is updated as new services are added.

The location of this file is defined in osc.properties. Set services.file equal to the path to this file.

Because this file contains environment-specific data, migration between environments should be manually reviewed. Typically, there will be different endpoint instances serving Dev, QA, Production, etc., and this file abstracts those concerns from the rest of the configuration.

Configuring the Template Store

The OSC can retrieve message templates from either the local file system or a database. Using a local file system is generally the faster, simpler option, but a database may be a better solution for multi-cluster machines, or where access to local storage is restricted.

File-Based Storage

To enable file-based storage:

1. Create a directory to hold the templates.
2. Add template.source=file to osc.properties
3. Add freemarker.templateDir equal to the directory from step 1 to the osc.properties.

Database-Based Storage

To enable database-based storage:

1. Create the table using the schema below
2. Create a JNDI bound JDBC resource named OSCDS that points to the database.
3. Add template.source=db to the osc.properties file.a.Optionally, configure template.minReadInterval to restrict how often the plugin will check for a new version of the template in the database. The value is given in milliseconds, and the default setting is 60000, i.e. 10 minutes.

OSC Database Schema

```
CREATE TABLE OSCTemplate (
TemplateID varchar2 (40) NOT NULL,
TemplateData CLOB NOT NULL,
```

```
UpdatedGMT timestamp NOT NULL,
primary key (TemplateID)
```

Basic File Layout

When using file-based template storage, most users will find the following to be the most convenient deployment structure.

OSC File Layout

/opt/Oracle/oipa (a.k.a. OIPA_Home)

| - osc

| - service - registry.xml

| - properties

| - | - osc.properties

| - templates

The properties directory is added to the JVM classpath entry. Templates are stored in the templates directory, the location of which is specified in the osc.properties file.

Configuring a Call

Configuring a call using the Outbound Service Connector involves three steps:

1. A service endpoint needs to be configured in the service registry.
2. A message template needs to be created.
3. OIPA needs to be configured to call the OSC extension and pass in data.

Service Definitions

A service endpoint identifies the technical details of where the message will be delivered and the template that should be used to create the message. This definition also contains the service ID that will be used elsewhere to identify the service.

There are several supported service types:

- SOAP services for invoking SOAP endpoints over HTTP
- file services for writing text to the file system
- Java Message Services for delivering text to JMS endpoints.

Each type of service requires different information to perform its function, but all services are defined in a common registry.

General Service Configuration

The root tag of the service-registry.xml file is <Services>. There are no XML namespaces used.

Each service is defined using a <Service> tag, a sub-element of <Services>. There are several elements common to every service definition.

Node	Required	Description
/Services/Service/@id	Yes	Used to identify service by the configuration. Must be unique.
/Services/Service/@type	Yes	Defines the service type. Valid values are soap, file, and jms.
/Services/Service/TemplateName	Yes	Defines the template key for this service.

SOAP Service

A SOAP service allows for the delivery of XML messages to a SOAP endpoint over HTTP. The associated template is responsible for defining the contents of the SOAP Body, while the service manages construction of the SOAP message itself in accordance with a provided WSDL.

The service definition is required to provide a WSDL, and to identify the service and port to be invoked. Several additional options allow for overriding default parameters.

Node	Required	Description
/Services/Service/WSDLLocation	Yes	Specifies the URL of the WSDL for the service.
/Services/Service/ServiceName	Yes	Specifies the name of the service (defined in the WSDL) to be called. Must be a valid QName.
/Services/Service/ServicePort	Yes	Specifies the port of the service (defined in the WSDL) to be called. Must be a valid QName.
/Services/Service/ServiceLocation	No	Overrides the service location in the WSDL with a specified value. Must be a valid URL.
/Services/Service/SOAPAction	No	Overrides the SOAP Action specified in the WSDL. See note below before including this tag.
/Services/Service/IgnoreResponse	No	Ignores the response from the service and returns success immediately. Valid options are true and false. The default is false.
/Services/Service/SecurityType	No	Can be used to override WS-Security or WS-Policy information.

Additional SOAP Considerations

The QName format is the text equivalent of the Java QName class. It specifies the Qualified Name of an element in a document. A QName is printed as {Namespace URI}Local Part. Also note that, in the case of a WSDL, the Namespace URI is the target namespace of the document, not the WSDL's namespace.

If a WSDL is not published for a service (or not available to the OIPA system online) it can be provided locally. Simply specify the WSDL URL location using a file:// prefix instead of http://.

SOAPActionIssues, the presence of the SOAP Action header in a SOAP message has meaning. Some services will expect to receive no header, some will expect a blank header, and some may expect a header with a value.For an absent SOAP Action header, neither the WSDL nor the service definition can include any reference to the header.For a blank SOAP Action header, provide the <SOAPAction> tag with no value in the service definition.In general, you can ignore this tag and omit it from service-registry.xml.

- **Waiting for a Response**: If the <IgnoreResponse> tag is used and has a value of true, then the OSC will not wait for a return value from the service. This means several things from an execution perspective:
- The plugin will still generate errors if there are issues opening a connection with the service or if a transmission error occurs while sending the message.
- The result of successful transmission will always be <Success/>.
- The plugin will not generate errors if the remote system was unable to understand or process the message.

There can be no assurance that the message was successfully received with a true value.

WebSphere-Specific Notes

Make sure the service name and port are identifying the correct namespace. The correct namespace is the XML namespace specified in the WSDL.

Add an empty SOAPACTION, since an empty SOAP action existed in the WSDL. The service being consumed may or may not have a SOAP action. Simply search for the SOAP action in the WSDL to see if one exists.

Modify the FreeMarker template to include the namespace in the template itself, as this gets injected directly into the body of the SOAP message.

It is helpful to use soapUI, as it gives the correct format of the SOAP packet.

It's recommended that TCPMon be used as an interceptor between the OSC extension and the service it is calling. This allows the user to inspect the SOAP packet that OSC is sending and compare it against the packet that soapUI is sending.

An example FreeMarker template that includes the XML namespace declaration:

```
<sayHello xmlns="http://osc.example.org/"/>
```

```
An example service-registry.xml file— notice the empty SOAPACTION tag:
```

```
<Service id="greetings" type="soap">
<TemplateName>Greetings.ftl</TemplateName>
<WSDLLocation>http://10.154.107.24:9999/osc/GreetingService.wsdl</WSDLLocation>
<ServiceName>{http://osc.example.org/\}GreetingService</ServiceName>
<ServicePort>{http://osc.example.org/\}GreetingServiceEndpointPort</ServicePort>
<SOAPAction/>
</Service>
```

File Service

A file service allows for the delivery of any textual data to the server's file system. In general, this should be leveraged with a remote file sharing protocol (NFS, Samba, etc.) or a scheduled FTP task to collate files between servers.

A service definition is required to provide a directory where files should be written. Several additional tags control the naming of files and replacement handling.

Node	Required	Description
/Services/Service/Directory	Yes	The directory where files will be written.
/Services/Service/FileNameType	No	The type of naming scheme to use. Valid options are Random, Reference, and Simple. The default value is Random.
/Services/Service/FileName	No	This element has no effect with the Random naming scheme. With Reference, it is the name of a variable in the context that holds the desired file name. With Simple, it is the name of the file.
/Services/Service/ReplaceExisting	No	If this element has a value of true, then a file of the same name as the requested output name will be overwritten. With

Node	Required	Description
		a value of false, an existing file of the same name will cause an error. The default is false.

Reports

Using a Simple file name and ReplaceExisting makes it easier to export batch reports to the file system as part of Plan or Company level processing. Don't forget to ensure output names are unique between reports.

JMS Service

A JMS service allows for the delivery of any textual data to a JMS-mapped destination (queue or topic).

A connection factory and destination are both required for delivery. Optional requirements include the QoS specifications for time to live and priority. These values should map to their relevant JMS values.

Node	Required	Description
/Services/Service/ConnectionFactory	Yes	The JNDI name of the JMS Connection Factory.
/Services/Service/Destination	Yes	The JNDI name of the JMS Destination.
/Services/Service/TimeToLive	No	The message life time in milliseconds (0–Unlimited).
/Services/Service/Priority	No	The priority of the message (Low: 0, High: 9, Default: 4).

Message Template

The message template is responsible for formatting the data appropriately for the type of service being used. The message must be text, as that is what is produced by the template engine. Certain service types may have additional requirements—namely, a web service endpoint will only accept messages consisting of valid XML.

The Outbound Services Connector operates using a template system for messages as opposed to XSLT. There are two key advantages to this approach. First, it avoids the overhead of building an intermediate XML format from the data. Second, it allows the template to interact directly with the data structures from OIPA.

Message templates are used to define the outgoing data payload. Payload formats differ depending on the type of delivery used. SOAP delivery requires a valid XML document. File and JMS delivery support any text format.

The message template language used in the Outbound Services Connector is FreeMarker. The FreeMarker Template Author Guide and Syntax Reference cover all of the functionality available.

The extension is responsible for exposing data to FreeMarker so it can be used in template processing. However, data is derived from the OIPA extension point in use. For the Math Transaction extension, all elements passed as parameters are exposed to the template by their NAME attribute. More information about passing data to the OSC can be found in the Extension Invocation section of this document.

Extension Invocation

Once a service is defined and the message template is constructed, the extension needs to be configured in OIPA. There are several configurable extension points in OIPA.

Transaction Math Extension (TYPE="PROCESS")

The math extension point relies on creating a new math variable of type PROCESS and setting certain values to initiate the extension call.

Node	Required	Description
MathVariable/@VARIABLENAME	Yes	The request variable name
MathVariable/@TYPE	Yes	Must be "PROCESS"
MathVariable/@NAMESPACE	Yes	Must be "com.oi.osc"
MathVariable/@OBJECT	Yes	Must be "MathPlugin"
MathVariable/@DATATYPE	Yes	Must be "OBJECT"

Once the call is configured, variables from the current execution can be passed into the plugin. It is important to note that the parameter name specified maps to the name used in the message template. Also, the Parameter element's text maps to the variable name in the transaction.

Node	Required	Description
MathVariable/Parameters/Parameter/@NAME	Yes	Defines the variable reference used in the template.
MathVariable/Parameters/Parameter/@TYPE	Yes	Valid values are INPUT and OUTPUT.
MathVariable/Parameters/Parameter	Yes	Defines the name of the MathVariable to be passed to the plugin.

OIPA Variable Datatypes

The message template language provides full support for interacting with complex objects. For this reason, it is highly recommended that arrays and maps be passed directly to the template for manipulation.

The OSC requires three specific parameters be defined in addition to those required for data mapping.

Parameter Name	Parameter Type	Parameter Value
service.id	INPUT	The name of service to be invoked. This value corresponds to the id attribute of the <Service> definition.
returnValue	OUTPUT	The name of a predefined TEXT variable to hold the return value from the service.

Parameter Name	Parameter Type	Parameter Value
errorValue	OUTPUT	The name of a predefined TEXT variable to hold any possible error value that may be returned from the service.

Output Values

If any error arises during processing, then the text of the error will be placed in errorValue. The configuration should check to ensure this variable is blank before processing is continued.

> **Tip :** One possible consideration to ensure an OSC call does not go through without the transaction being successfully processed (transaction integrity requirement) is to consider spawning a separate transaction whose primary purpose is to call OSC. This will ensure all post-execution details from the transaction are also available on the OSC call.

Caching

The OSC leverages two kinds of caching during execution: a local cache and a distributed cache. Both caches operate through the same interface, but are semantically different. The local cache supports non-serializable objects, whereas the distributed cache does not.

By default, the OSC uses a local LRU cache and Coherence as a distributed cache. If Coherence support is not desired, then the LRU cache can be substituted for the distributed cache as well.

Configuring Coherence

The Coherence configuration for OSC needs to be merged with the configuration for OIPA. You can do this by adding the following to the coherence-cache-config.xml file.

Coherence Configuration for OSC

```
<!-- Map Templates to Distributed Near Scheme -->
<caching-scheme-mapping>
    <cache-mapping>
        <cache-name>OSC_Templates</cache-name>
        <scheme-name>OSCScheme</scheme-name>
    </cache-mapping>
</caching-scheme-mapping>
<!-- OSC Distributed In-memory Cache -->
<caching-schemes>
    <local-scheme>
        <scheme-name>SampleMemoryScheme</scheme-name>
    </local-scheme>
        <distributed-scheme>
            <scheme-name>OSCScheme</scheme-name>
                <backing-map-scheme>
                    <local-scheme>
                        <scheme-ref>SampleMemoryScheme</scheme-ref>
                    </local-scheme>
                </backing-map-scheme>
        </distributed-scheme>
</caching-schemes>
```

This configuration can be further adjusted to meet a deployment's specific needs.

Configuring the LRU Cache

The LRU cache is a simple, built-in solution for caching non-serializable objects. It is configured entirely through the osc.properties file.

Property Name	Value Type	Default	Notes
cache.lru.<cache_name>.initSize	Integer	100	For a given LRU sub-cache, specify the initial size
cache.lru.<cache_name>.maxSize	Integer	1000	For a given LRU sub-cache, specify the maximum size.

Where the <cache_name> is the internal sub-cache ID (currently only OSCJMS and OSC_Templates).

Disabling the Cache

One or both of the caches can be disabled by overriding the default cache implementation. A special no-op cache has been provided, which stops the cache from using any data.

To change to the **no-op cache**, set **cache.distributed.type** or **cache.local.type** to **com.oi.osc.cache.NoCache**.

Examples

Push Notification Scenario

Company Foo wants to push information to an external service every time a new policy is submitted. The service expects an XML file like:

Sample XML

```
<PolicyNotify>
<PolNum>Policy Number</PolNum>
<Amount>Policy Face Amount</Amount>
</PolicyNotify>
```

Service Definition

First, define the service in the service-registry.xml.

Service Registry Definition

```
<Service id="policyNotify" type="soap">
    <TemplateName>PolicyNotify.ftl</TemplateName>
    <WSDLLocation> http://www.foo.com/services/PolicyNotify?wsdl</WSDLLocation>
    <ServiceName>{{+}ht-
    tp://www.foo.com/xml/PolicyNotify/+}PolicyNotifyService</ServiceName>
    <ServicePort>{{+}ht-
    tp://www.foo.com/xml/PolicyNotify/+}PolicyNotifyServicePort</ServicePort>
</Service>
```

This defines a service with the policyNotify ID as a SOAP service. The WSDL specified must be accessible to the application server at runtime. The service name and port are taken from the WSDL and identify the component to be invoked.

Template Definition

Second, define the message template.

Message Template - PolicyNotify.ftl

```
<PolicyNotify>
<PolNum>${PolicyNumber}</PolNum>
<Amount>${FaceAmount}</Amount>
</PolicyNotify>
```

The template name specified in the <TemplateName> tag of the service description must match the name of the template file.

Since this is a template, the file should closely resemble the downstream format. Variables are introduced where content from OIPA is needed.

Extension Invocation

Third, call the extension from the OIPA configuration.

OIPA Transaction Configuration

```
<MathVariable VARIABLENAME="ServiceID" TYPE="VALUE" DATATYPE-
E="TEXT">policyNotify</MathVariable>
<MathVariable VARIABLENAME="ReturnValue" TYPE="VALUE" DATATYPE-
E="TEXT"></MathVariable>
<MathVariable VARIABLENAME="ErrorValue" TYPE="VALUE" DATATYPE-
E="TEXT"></MathVariable>
<MathVariable VARIABLENAME="PolicyNotify" TYPE="PROCESS" NAMESPACE="com.oi.osc"
OBJECT="MathPlugin" DATATYPE="OBJECT">
<!-- Required Parameters for every call -->
<Parameter NAME="service.id" TYPE="INPUT">ServiceID</Parameter>
<Parameter NAME="returnValue" TYPE="OUTPUT">ReturnValue</Parameter>
<Parameter NAME="errorValue" TYPE="OUTPUT">ErrorValue</Parameter>
<!-- Data Parameters -->
<Parameter NAME="PolicyNumber" TYPE="INPUT">PolicyNumber</Parameter>
<Parameter NAME="FaceAmount" TYPE="INPUT">FaceAmount</Parameter>
</MathVariable>
```

This sample configuration defines three new text variables to hold values for the call: one for the service ID and two for the return values. It assumes that PolicyNumber and FaceAmount are already defined prior to this definition.

Alternative Delivery—File

If Company Foo wants to leverage file delivery instead of SOAP, then the only configuration that needs to change is the service definition.

Policy Notify - File

```
<Service id="policyNotify" type="file">
    <TemplateName>PolicyNotify.ftl</TemplateName>
    <Directory>/opt/interfaces/policyNotify</Directory>
    <FileNameType>Reference</FileNameType>
    <FileName>PolicyNumber</FileName>
</Service>
```

This specifies the directory that should be used for output as well as some optional settings. The <FileNameType> element specifies that the filename is a reference to a provided value, in this case PolicyNumber. Since the directory is specific to the interface this should be alright. ReplaceExisting could also be added to allow for overwriting the file should the activity be reprocessed.

Alternative Delivery—JMS

If Company Foo wants to leverage JMS delivery instead of SOAP, then the only configuration that needs to change is the service definition.

Policy Notify - JMS

```
<Service id="policyNotify" type="jms">
    <TemplateName>PolicyNotify.ftl</TemplateName>
    <ConnectionFactory>jms/ConnectionFactory</ConnectionFactory>
    <Destination>jms/Destination</Destination>
</Service>
```

JMS delivery only requires a ConnectionFactory and Destination reference. These are both JMS artifacts that are configured in the application server and bound to a JVM or cluster. If the destination is shared between interfaces, then consider setting the Priority flag to rank messages.

Appendix

Appendix 1—OSC Property File Values

The following table is a complete list of all the valid property keys for the osc.properties file. Full descriptions of these properties and their meanings can be found elsewhere in this document. In most cases, default values should not need to be changed unless they are required.

Property Name	Value Type	Required	Default	Notes
cache.distributed.type	Class implementing Cache	No	com.oi.osc.cache.CoherenceCache	
cache.local.type	Class implementing Cache	No	com.oi.osc.cache.LRUCache	
cache.lru.<cache_name>.initSize	Integer	No	100	For a given LRU sub-cache, specify the initial size
cache.lru.<cache_name>.maxSize	Integer	No	1000	For a given LRU sub-cache, specify the maximum size.
factory.constructor	Class implementing IConstructor	No	com.oi.osc.constructor.FreeMarkerConstructor	
factory.security	Class implementing ISecurityProvider	No	com.oi.osc.security.WalletSecurityProvider	
freemarker.templateDir	String	Maybe	n/a	Required if loading templates from filesystem. Must specify full path to template directory.
services.file	String	Yes	n/a	Must specify full path to service registry

Property Name	Value Type	Requir-ed	Default	Notes
tem-plate.minReadIn-terval	Integer	No	60000	Value in mil-liseconds
template.source	{ file, db }	Yes	n/a	User must specify how templates will be loaded
wallet.path	String	Maybe	n/a	If using the Wal-letSe-curityProvider, this must specify a path to the Oracle Wallet file
wallet.key	String	Maybe	n/a	If using the Wal-letSe-curityProvider, this must specify the wallet password

Appendix 2—Service Registry Tags

The following table is a complete list of all the possible service registry tags. Full descriptions of these tags and their meaning can be found elsewhere in this document.

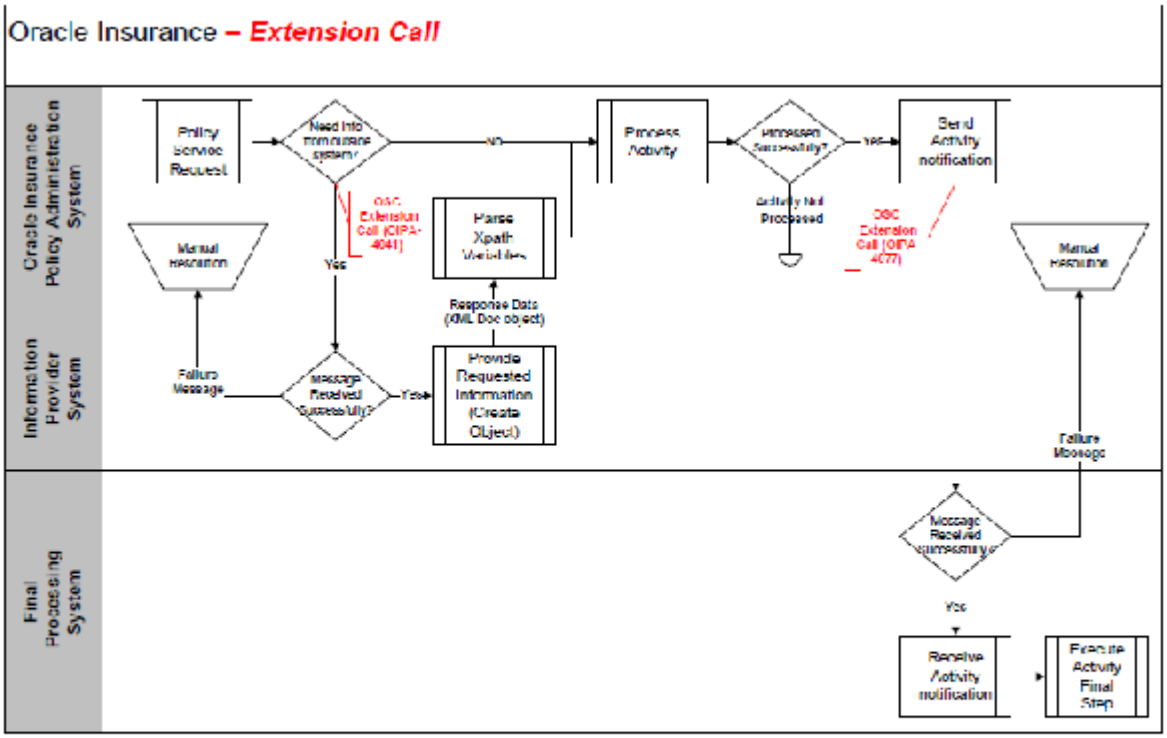
All Node listings are relative to <Services>/<Service>.

The "Relevance" column" indicates when the tag is relevant to a service definition. The "Required" column then indicates when whether the tag is required in that particular case.

Node	Value Type	Relevance	Required	Notes
@id	String	All	Yes	Must be unique to the service registry
@type	{ soap, file, jms }	All	Yes	
TemplateName	String	All	Yes	Must match a valid template file name or data-base key
WSDLLocation	URL	SOAP	Yes	URL of the SOAP service's WSDL
ServiceName	QName	SOAP	Yes	QName of the service to invoke
ServicePort	QName	SOAP	Yes	QName of the service port to invoke

Node	Value Type	Relevance	Required	Notes
ServiceLocation	URL	SOAP	No	Overrides value in WSDL
SOAPAction	String	SOAP	No	Do not include this tag unless specifically required
IgnoreResponse	Boolean	SOAP	No	
Username	String	SOAP	No	
PasswordKey	String	SOAP	No	
SecurityType	{ HTTP, XWSS }	SOAP	No	
XWSSConfig	String	SOAP: XWSS	Yes	Path to XWSS policy file to apply to service
Directory	String	File	Yes	Absolute path to directory for files
FileNameType	{ Random, Reference, Simple }	File	No	Defaults to Random
FileName	String	File: reference, simple	No	Specifies a reference to a variable with the file name, or a reference to the file name itself, depending on FileNameType.
ReplaceExisting	Boolean	File: reference, simple	No	Specifies whether to overwrite an existing file. Default is false.
ConnectionFactory	String	JMS	Yes	JNDI name of JMS Connection Factory
Destination	String	JMS	Yes	JNDI name of JMS Destination
TimeToLive	Integer	JMS	No	Time to live, in seconds. 0 = unlimited. Default is 0.
Priority	Integer [0-9]	JMS	No	Message priority, low to high. Default is 4.

Appendix 3—OSC Extension Call Process



Assumptions:

1. SOAP messaging type
2. Normal system messages for failed SOAP messages apply
3. Activity will not process without information (failed message or failed XML Document creation)
4. Configuration is expected to handle parsing Xpath issues with defaults
5. Activity not completed is left pending
6. Failure message is stored in table for future process