# Table of Contents

Oracle BPM Web Services Best Practices document ............................................. 3

Consuming External Web Services from Oracle BPM ........................................... 4
  Introspecting Web Services ........................................................................... 4
  Changing the WSDL URL at runtime ............................................................. 8
  Discovering Web Services over HTTPS using certificates ............................. 8

Web Service Authentication .............................................................................. 11
  WS-Security: UserName Token Profile secured Web Services .................. 11
  HTTP Basic Authentication (Transport security) ......................................... 13
  Specifying a different User/Password when invoking the Web Service using PBL .......................................................... 14

Exposing OBPM assets as Web Services for external consumption ............... 16
  Specifying Security for Processes exposed as Web Services ................... 16
  Specifying Security for PAPI-WS ............................................................... 17

Client Frameworks to connect to Process as a Web Service and PAPI-WS ....... 18
  Client Authentication ................................................................................. 18
    Session Id based Client Authentication ..................................................... 18
    WS-Security UserName Token Profile authentication .............................. 19
  PAPI-WS Single Sign On ......................................................................... 20

General Best Practices ..................................................................................... 22
  Web Service Versioning ............................................................................. 22

Web Service Exception Handling .................................................................... 22
Oracle BPM Web Services Best Practices document

The following document will concentrate on providing general information about how to use Web Services with Oracle BPM.

The document will be divided in the following sections:

a) Consuming External Web Services from Oracle BPM
b) Exposing OBPM assets as Web Services for external consumption
c) Best Practices
d) Troubleshooting
Consuming External Web Services from Oracle BPM

This section of the document will concentrate on describing the different procedures and details about how to consume and discover a Web Service that can be introspected and then invoked from an Oracle BPM business process.

Introspecting Web Services

Oracle BPM has the capability of discovering or introspecting a Web Service WSDL Definition using a URL. Typically, the WSDLs are exposed through an HTTP(S) accessible URL and in the case of the PayPal Web Services, these are the available URLs:

- http://www.paypal.com/wsd1/PayPalSvc.wsdl
- https://www.paypal.com/wsd1/PayPalSvc.wsdl

If we use the Oracle BPM Studio Web Service Introspector Wizard we can discover either one of these services with the following wizard sequence:
We can right click on the Studio Catalog Module and select “Catalogue Component -> Web Service” as shown in the figure above.

The next step would be to provide the WSDL URL as shown in the figure below.

Or alternatively use the WSDL URL using the HTTPS transport protocol as shown below:
Click “Next” to continue and you will see the progress of the introspection going on as shown in the figure below.

Click “Finish” to complete the introspection. After this step, you are ready to use the discovered and introspected Web Service components.

The result of discovering or introspecting a Web Service also defines 2 new entries in the project “External Resources” section as shown in the following 2 figures below.
In this case, we are selecting the Web Service one specifically and it pointing to the Web Service End Point “/2.0/”. It is also making reference to a specific Server Configuration that has the machine location as shown in the figure below.

In this figure, you will see the URL textfield containing the address or URL to the Web Service EndPoint extracted from the introspected or discovered WSDL.
Changing the WSDL URL at runtime

Once a project has been published and deployed, it is possible to change the values for the previously mentioned External resources: Web Service EndPoint Host and Port as well as the EndPoint URL in that EndPoint machine location.

This can be done through the Process Administrator Web Interface as shown below.

Discovering Web Services over HTTPS using certificates

When trying to discover a Web Service WSDL over HTTPS that uses certificates, if the client does not have the certificate installed on the client application (in this case Oracle BPM Studio), there will be problems in reaching out to it and properly addressing the introspection procedure.

The typical error message with this case when the certificate is not installed in the Oracle BPM Studio would be something like this:

Introspecting...
downloading /stockquote.wsdl
[Error] I/O error: sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target..
[Error] Instrospection exception:
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:

Oracle – Proprietary and Confidential
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target.

This means the Oracle BPM Studio application is receiving a certificate from the Web Server hosting the WSDL that it cannot trust. As such, we need to install the certificate on the Java Virtual Machine used by Oracle BPM Studio. Below we will describe these steps.

1. Copy the client certificate (file with .cer extension) to the machine where Oracle BPM Studio is installed. For example c:\CitiCertificate.cer.

2. Install/Import the certificate on a keystore that is being used by the Oracle BPM Studio. By default, Oracle BPM Studio does not use a KeyStore, so we will most likely need to generate a new one. This can be done with a command line as follows:

```bash
> $JAVA_HOME/bin/keytool -import -alias CitiGroup -file c:\CitiCertificate.cer -keystore c:\OBPMStudioKeyStore.store -storepass password
```

In this case, we assume that the Organization name for which the certificate was extended is CitiGroup. We also assume that the password for the keystore in which we will import the certificate will be password. We also assume you have a Java Developer Kit (JDK) installed since these certificate related commands are not distributed with the JVM that comes with Oracle BPM Studio.

3. Set the following properties in the following files depending the Oracle BPM Studio version used:

**ALBPM Studio 5.7:** Edit the $STUDIO/bin/albpmstudio.properties and add the following 2 system properties:

- `javax.net.ssl.trustStore=c:/CitiKeyStore.store`
- `javax.net.ssl.trustStorePassword=password`

as indicated in the line below:

```java
# fuego.boot.verbose=true
fuego.log.runningtime=true
fuego.update.disabled=true
javax.net.ssl.trustStore=c:/CitiKeyStore.store
javax.net.ssl.trustStorePassword=password
#
```

**ALBPM Studio 6.0/OBPM Studio 10gR3:** Edit the $STUDIO/eclipse/eclipse.ini and add the following system properties:

Oracle – Proprietary and Confidential
• javax.net.ssl.trustStore=c:/CitiKeyStore.store
• javax.net.ssl.trustStorePassword=password

as indicated in the line below:

...
-Djava.endorsed.dirs=""
-XX:PermSize=64M
-Djavax.net.ssl.trustStore=c:/CitiKeyStore.store
-Djavax.net.ssl.trustStorePassword=password

IMPORTANT: It is worth pointing out that the properties are NOT the ones with the prefix “keyStore”, but the ones with the prefix “trustStore”.

4. Launch Oracle BPM Studio again and this time, when you try to instrospect the Web Service it should work.

Another alternative path for this is to download the WSDL file with the browser locally and instrospect the Web Service from the filesystem directly.
Web Service Authentication

This subsection of the document will concentrate on addressing how Oracle BPM can invoke secured web services.

WS-Security: UserName Token Profile secured Web Services

The UserName Token Profile authentication is not an out of the box feature in ALBPM 5.7. This particular implementation following the WS Username Token Profile 1.1 is provided starting in ALBPM 6.0 and also in OBPM 10gR3.

The following section will describe how to introspect a Web Service that at runtime uses WS-Security UserName Token Profile. This will be illustrated in Oracle BPM Studio 10gR3 but it is the same in ALBPM Studio 6.0.

1. Launch the Web Service introspection wizard as shown below.

2. Click “Next” to continue and wait until the introspection completes as shown below. Click “Finish” to proceed.
3. After the Web Service has been introspected, it is necessary to go to the Web Service generated External Resources as shown below.

4. You will see the “Security” Tab at the bottom of the panel. Click on it and you will have the following panel as shown in the figure below.
5. If the Web Service is not using WS-Security UserName Token Profile, the value in the “Send UserName Token” drop down should be “None”. Otherwise, if the Web Service is protected with WS-Security UserName Token Profile, select the way the credentials will be sent from the Web Service itself. The options are to send the credentials in plain format or encrypted with a Digest.

6. Select the appropriate mechanism for sending the UserName Token and specify the default username and password for properly authenticating the future Web Service calls.

HTTP Basic Authentication (Transport security)

In addition to the security enforced at the SOAP stack level, it is possible that the Web Container using Basic Authentication protects the Web Service exposed through HTTP at the transport level.

Oracle BPM started supporting HTTP Basic Authentication in ALBPM 6.0 and it is also supported in OBPM 10gR3.

Instead of configuring this security aspect in the Web Service External Resource, it is necessary to specify it in the “Server Configuration” external resource.
From the figure above, it should be clear that the “Requires HTTP Basic Authentication” check box is selected and that valid values are specified in the User/Password TextField in this panel.

**Specifying a different User/Password when invoking the Web Service using PBL**

It is possible that the Web Service calls may not need to be done using the general credentials provided through the procedure below. In the event that the call needs to be impersonated with a different username and password, this is the PBL code that can be written for this particular use case. The following sample assumes that the Web Service is protected both at the SOAP and HTTP levels. You will need to
adapt your code based on your needs but we wanted to present all options in this code snippet.

```csharp
configuration = Fuego.WebServices.Configuration();
// Add Endpoint
configuration.endpoint = endpoint;
// Setting Http Basic Authentication
endpoint.setUsername("username");
endpoint.setPassword("mypassword");
// Setting UsernameTokenProfile-Digest
test1","password");
usernameTokenPlain = Fuego.WebService.UsernameTokenProfileSecurityPolicy("t
est2","password");
policies as Fuego.WebService.SecurityPolicy[];
policies[]\=usernameTokenPlain;
// Add policy
configuration.securityPolicies = policies;
service = Module.Pepe.Test1Service(configuration);
test1 service
   using parameter = null
      returning parameterOutput = parameterOutput
```

Alternatively, you can get the External Resource definition by using the following Configuration object constructor by providing the case sensitive name of the External Resource name as the single constructor argument:

```csharp
configuration =
Fuego.WebServices.Configuration("MyExternalResources")
;"
Exposing OBPM assets as Web Services for external consumption

This section of the document will concentrate on details as to how expose certain Oracle BPM assets as Web Services. Within this category of assets will find:

a) Processes Exposed as Web Services
b) PAPI-WS

Specifying Security for Processes exposed as Web Services

When exposing a process as a Web Service, it is possible to specify how it is going to be secured. At design time and in Oracle BPM Studio 10gR3 (and also ALBPM 6.0), it is possible to specify what is the Web Service security mechanism for doing this.

In the Studio IDE, go through the following menu path: Process > Process Web Service. A panel similar to the one below will be presented where it is possible to specify the methods that will expose the process public interface and where it is also possible to specify the security.

![Panel for specifying security](image)
In the figure above, it is possible to specify WS-Security UserName Token Profile as well as HTTP Basic Authentication as indicated in the previous sections.

When the project is published and deployed, proper External Resources will be generated for every process exposed as a Web Service and the right credentials can be specified there.

**Specifying Security for PAPI-WS**

Another application that can leverage the WS-Security and HTTP Basic Authentication is PAPI-WS. As it may be known, starting in ALBPM 6.0 (and also as part of OBPM 10gR3), PAPI-WS is bundled as a single Web Application that can server SOAP requests to interact with business processes with a finer granular control than Processes exposed as Web Services.

When launching the Admin Center component of any of the Enterprise distributions, it will be possible to enter the “Configuration” of this installation and navigate to the PAPI-WS Tab as shown in the figure below.
In this panel, it is possible to select the “Enable HTTP Basic Authentication” (disabled by default) as well as “Enable UserName Token Profile Authentication” (enabled by default).

It is then responsibility of the client applications to provide proper credentials for the level of security requested by the PAPI-WS Web Application.

**Client Frameworks to connect to Process as a Web Service and PAPI-WS**

When a process interface has been created for a given process or you want to connect to the Oracle BPM PAPI-WS API, the following 2 Web Service stack frameworks have been validated:

- JAX-WS 2.1.x
- .NET Framework 2.0.5 and Microsoft Web Service Enhancements 3.0.

There have been some problems when testing different Web Service client frameworks like AXIS. So AXIS is not recommended for invoking Web Service components in Oracle BPM.

Sample code snippets can be found through this link: [http://download.oracle.com/docs/cd/E13154_01/bpm/docs65/papi/index.html](http://download.oracle.com/docs/cd/E13154_01/bpm/docs65/papi/index.html). Check the “Oracle BPM PAPI Web Service” section for the data.

**Client Authentication**

When you are implementing your own Web Service client application to talk to a business process exposed as a Web Service you have the following alternatives:

- Session Id based authentication
- WS-Security UserName Token Profile authentication

Let’s see in details each in the next 2 sub-sections.

**Session Id based Client Authentication**

This authentication mechanism is the only one available in ALBPM 5.5 and ALBPM 5.7. It is also supported in deprecated fashion in ALBPM 6.0 but if you are...
planning on using this new release, we recommend the use of WS-Security based authentication.

In this case, the client first needs to create a session and then use the returned session id in all subsequent calls. The client application is also in charge of the session life cycle and it is the responsibility of the client application to close the session when it is no longer needed. It is worth to mention that the session creation is not a quick operation since it needs to also identify permissions so we encourage session caching techniques on the client side to prevent opening and closing sessions too frequently.

The following is a snippet of code on how to create the session with PAPI-WS using the Session Id mechanism:

```java
...
ProcessService service = locator.getProcessService();
String userId = "user";
String password = "password";
String mySessionId = "";
mySessionId = service.createSession(userId, password);
InstanceInfo[] instances = service.getInstancesByView(mySessionId, view);
...
service.closeSession(mySessionId);
...
```

**WS-Security UserName Token Profile authentication**

This authentication mechanism has been introduced in ALBPM 6.0 and only available in ALBPM 6.0 and OBPM 10gR3. It replaces the session id based authentication and the credentials as defined in the WS-Security specification definition can travel in the SOAP message.

In this case, the client application does not need to do session management as this is automatically managed on the server side by the PAPI-WS application of the BPM Engine providing processes exposed as Web Services. Usually these
sessions will leave for a configurable amount of time and when they are not already created, they will be created on the flight automatically.

You can find code snippets for creating an authentication request through this link:


**PAPI-WS Single Sign On**

PAPI-WS functionality is provided through the implementation of a Web Application. In ALBPM 5.5 and ALBPM 5.7, the PAPI-WS functionality was jointly distributed with the ALBPM Portal Web Application. In ALBPM 6.0 and OBPM 10gR3, the PAPI-WS functionality was decoupled from the ALBPM Portal or OBPM WorkSpace Web Application implementation to separate and configure these individually. One of these things is security.

In ALBPM 5.5 and ALBPM 5.7, it is possible to configure the Directory Service using the Trusted mode where at authentication or login time, only the identity of the connected user is validated without a password matching validation. This functionality also exists in ALBPM 6.0 and OBPM 10gR3. This configuration is inherited by configuring the Directory Service in Trusted mode. A set of particular preferences needs to be defined in the directory.properties (ALBPM 5.5 and ALBPM 5.7) or directory.xml (ALBPM 6.0 and OBPM 10gR3) as well as some specifics in the Directory Service Database. This particular configuration is out of the scope of this document, but it is included here as a reference of available functionality.

In addition to the Trusted Directory Service mode, ALBPM 6.0 and OBPM 10gR3 can implement a SSO module so additional validation is available. This SSO can be seen in the Admin Center Tab. The settings here are then propagated to the papiws.properties file in the following property entries:

- fuego.papiws.enableSSO
- fuego.papiws.sso.loginClassName

Below you will find the reference implementation for a custom implementation of the loginClassName.

```java
public class SSOUserLogin
    implements SSOUserLoginInterface
{
    //~ Methods ..............................................................................................
```
public String getUser(final HttpServletRequest request, final HttpServletResponse response)
    throws SSOLoginException
{
    return request.getRemoteUser();
}

public String getPassword(final HttpServletRequest request, final HttpServletResponse response)
    throws SSOLoginException
{
    return null;
}

public boolean skipFDIAuthentication()
{
    return true;
}

public String getLogoutURL()
{
    return null;
}

public String getLogoutRelativePath()
{
    return null;
}

In the previous sub-sections we have identified how to create sessions with a sessionId and WS-Security authentication mechanisms. If you are configuring SSO for PAPI-WS, then you will still need to create the sessions through the provided mechanisms and it will depend on the SSO authentication module what to do with the provided data. The default mechanism implies checking the participant exists in the Oracle BPM Directory Service but no password matching is exercised. So the password provided when creating a session does not need to have a valid one.
General Best Practices

This section of the document will mainly concentrate on providing general best practices when working with Web Services mostly when they are consumed from within Oracle BPM through the introspection framework.

**Web Service Versioning**

It would not be uncommon to see the Web Service evolve over time. Generally speaking you can have a Web Service add new methods, change arguments on an existing methods to just name a few of the most common upgrade/maintenance operations.

ALBPM does actually version the processes as well as the catalog it uses all together, but one of the problems with Web Services is that possible different implementation versions may use the same External Resource.

The best practice in this case is that if for some reason, the interfaces of the Web Service changes and you have in flight instances that may still need to invoke this Web Service, that the Web Service is introspected again on a different Module. This will force a new External Resource different than the one used by the previous version of the Web Service used by previous versions or revisions. This will prevent the undesired effect of an existing old instance trying to invoke a new interface that is not compatible.

**Web Service Exception Handling**

The Oracle BPM Engine conceptually manage 2 different type of exceptions:

- **System Exceptions:** These are typically associated to unexpected component invocation behaviors not related with business logic that can be checked in the implementation of your business process or BPM Objects. For example, the component backend not being available, network issues, etc. These type of exceptions will be

- **Business Exceptions:** These are typically defined within the scope of the BPM Project Catalog and are triggered manually by the developer in the code they implement. For example, we may check the different pieces of a mail address and find that the zip code does not match the address. In
In this case, we can manually raise a Business Exception due to this coded validation.

Depending on what is the exception type, the Oracle BPM Engine will react differently to them. It is imperative that the differences are known as well as the effects they have on exception catching/trapping coding practices.

In ALBPM 5.7, the invocation of a Web Service method can trigger both type of exceptions. When the Web Service responds with a SOAP fault handler, it will be masqueraded as a Business Exception. On the other hand, specific underlying Web Service problems will be trapped as a System Exception. All Web Service Exceptions must be treated as System Exceptions from an exception handling perspective. If it is necessary to change the exception mechanism behavior, it will mandatory to re-catalog the whole Web Service again since new metadata needs to be refreshed and updated. It is also necessary to generate a new revision of the project if it is already deployed and up and running.

Whether the Web Service exceptions are System or Business can be defined in the “Advanced” Tab of the Web Service definition as shown in the figure below. Select the checkbox appropriately.
When SOAP Faults are declared in the WSDL, these will be represented as Exceptions in the Web Service module that has been introspected. These exceptions would then be able to be used in any exception catching construct as shown below:

```java
do
   WebService.callMethod(m1, m2)
on e as MySOAPFaultException
    // Do Something with this Web Service Exception
end
```

In the event that the exceptions are NOT defined as SOAP Fault Handlers, it will be necessary to use an exception catcher using the Any type as shown below:

```java
do
   WebService.callMethod(m1, m2)
on e as Any
    // Do Something with this Web Service Exception
end
```

Something worth pointing out is that SOAP Fault exceptions are not Java Exceptions and as such, they do not extend from the java.lang.Exception superclass. Any is a exception layer on top of java.lang.exception (in the PBL scripting language), that allows handling these non-declared SOAP Faults.
Troubleshooting

While discovering Web Services accessible over the network, it is possible that a proxy needs to be specified. If the Web Service WSDL cannot be accessed as shown in the figure below, it is necessary to specify the proxy host and port as system properties for the Java Virtual Machine running the introspection as well as for the one running the BPM Engine.

The system properties that need to be specified in the Oracle BPM Studio and Oracle BPM Engine are (they are case sensitive):

- http.proxyHost
- http.proxyPort

These can be set in the Oracle BPM Studio 5.7 through the following menu paths: “File” > “Preferences” > “Connection Settings” as shown in the figure below.
According to Sun’s specification, if the transport protocol used is HTTPS instead of HTTP, the system properties are (note the “s” in the prefix word):

- `https.proxyHost`
- `https.proxyPort`

These properties cannot be added through the Studio UI and they need to be added to the Oracle BPM Studio 5.7 Launcher properties file under `$STUDIO/bin/albpmstudio.lax` as shown in the screenshot below.

The important line should read as follows:

```
lax.nl.java.option.additional=-ea -Xms64m -Xmx256m -Dhttps.proxyHost=proxyHost.oracle.com -Dhttps.proxyPort=8080 -Dfile.encoding=UTF-8
```

In ALBPM 6.0, these properties can be specified in the `$STUDIO/eclipse/eclipse.ini` file.

In Oracle BPM Studio 10gR3, the properties in the `$STUDIO/eclipse/eclipse.ini` does not work, and these proxy settings need to be specified in the IDE itself following this menu path: "Window" -> "Preferences" ->
"General" -> "Network Connections". In this panel, we need to specify the HTTP and HTTPS proxy servers. Check the figure below.

After these changes, restart Oracle BPM Studio and try the Web Service introspection again.