1. TABLE OF CONTENTS

[SOA introduction 2](#_Toc510019557)

[1.1 Configuring SSL 2](#_Toc510019558)

[1.1.1 Using SSL Certificates When the SOA/BPM Server Is Configured with an HTTPS Port 2](#_Toc510019559)

[1.1.2 Recommendation to Configure Either All or No Managed Servers with SSL 3](#_Toc510019560)

[1.1.3 Switching from Non-SSL to SSL Configurations with Oracle BPM Worklist 3](#_Toc510019561)

[1.1.4 Configuring SOA Composite Applications for Two-Way SSL Communication 4](#_Toc510019562)

[1.1.5 Invoking References in One-Way SSL Environments in Oracle JDeveloper 6](#_Toc510019563)

[1.1.6 Configuring Oracle SOA Suite and Oracle HTTP Server for SSL Communication 6](#_Toc510019564)

[1.1.6.1 Configuring Oracle HTTP Server for SSL Communication 6](#_Toc510019565)

[1.1.6.2 Configuring Certificates for Oracle Client, Oracle HTTP Server, and Oracle WebLogic Server 8](#_Toc510019566)

[1.1.7 Configuring SSL Between SOA Composite Application Instances and Oracle WebCache 8](#_Toc510019567)

[1.1.8 Using a Custom Trust Store for One-Way SSL During Design Time 9](#_Toc510019568)

[1.1.9 Enabling an Asynchronous Process Deployed to an SSL-Enabled, Managed Server to Invoke Another Asynchronous Process Over HTTP 9](#_Toc510019569)

# SOA introduction

A **service-oriented architecture** (**SOA**) is a style of software design where services are provided to the other components by application components, through a communication protocol over a network.

### Configuring SSL

This section describes configuring secure socket layer (SSL) in Oracle SOA Suite and Oracle Business Process Management environments.

#### Using SSL Certificates When the SOA/BPM Server Is Configured with an HTTPS Port

If the SOA/BPM server is configured with an HTTPS port, ensure that your SSL certificate adheres to the following standards:

* The certificate that the server presents to SSL clients (the browser or other internal clients such as the notification senders) is a trusted certificate by its own trust store (the CA store).
* If the certificate for the server is self-signed, ensure that you add it to the trust store. That is, the certificate that the server presents must validate itself against the server trust store.

Not doing so can cause problems when task notifications are sent. For example, you can receive the error message shown in Example in the server out log (for example, soa\_server1.out).

***Example 5-1 Task Notification Error***

<Sep 13, 2011 12:59:41 AM PDT> <Error> <oracle.soa.services.workflow.common>

<BEA-000000> <<.>

ORABPEL-0

at

oracle.bpel.services.workflow.task.notification.

TaskNotifications.getEmailPaylOad

(TaskNotifications.java:1354)

at

oracle.bpel.services.workflow.task.notification.

TaskNotifications.getEmailNotificationContent

(TaskNotifications.java:987)

at

weblogic.jms.client.JMSSession$UseForRunnable.run

(JMSSession.java:5170)

at

weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl.run(SelfTuningWorkManagerI

mpl.java:528)

at weblogic.work.ExecuteThread.execute(ExecuteThread.java:209)

at weblogic.work.ExecuteThread.run(ExecuteThread.java:178)

Caused By: javax.net.ssl.SSLKeyException: [Security:090477]Certificate chain

received from adcgen15.us.oracle.com - 10.232.152.78 was not trusted causing

SSL handshake failure.

at

com.certicom.tls.interfaceimpl.TLSConnectionImpl.

fireException(UnknownSource)

at

com.certicom.tls.interfaceimpl.TLSConnectionImpl.

fireAlertSent(UnknownSource)

at

com.certicom.tls.record.handshake.HandshakeHandler.

fireAlert(Unknown Source).

ficationContent(TaskNotifications.java:987)

at

weblogic.jms.client.JMSSession$UseForRunnable.run(JMSSession.java:5170)

at

weblogic.work.SelfTuningWorkManagerImpl$WorkAdapterImpl.run(SelfTuningWorkManagerI

mpl.java:528)

at weblogic.work.ExecuteThread.execute(ExecuteThread.java:209)

at weblogic.work.ExecuteThread.run(ExecuteThread.java:178)

Caused By: javax.net.ssl.SSLKeyException: [Security:090477]Certificate chain

received from adcgen15.us.oracle.com - 10.232.152.78 was not trusted causing

SSL handshake failure.

at

com.certicom.tls.interfaceimpl.TLSConnectionImpl.

fireException(UnknownSource)

at

com.certicom.tls.interfaceimpl.TLSConnectionImpl.

fireAlertSent(UnknownSource)

at

com.certicom.tls.record.handshake.HandshakeHandler.

fireAlert(Unknown Source

#### Recommendation to Configure Either All or No Managed Servers with SSL

As a best practice, Oracle recommends that you configure either *all* managed servers or *no* managed servers with SSL (SOA, BAM, and so on). Configuring some managed servers with SSL, while not configuring others, may lead to undesirable results in Oracle BPM Worklist and Oracle Web Services Manager (OWSM). For example, if there is an SSL-configured, managed server (bam\_server), servers not configured with SSL are not used by OWSM. In cases in which an SSL-configured server is down, it causes OWSM to be in a down state, which in turn causes Oracle BPM Worklist to be in a down state.

#### Switching from Non-SSL to SSL Configurations with Oracle BPM Worklist

Switching from non-SSL to SSL configurations with Oracle BPM Worklist requires the **Frontend Host** and **Frontend HTTPS Port** fields to be set in Oracle WebLogic Server Administration Console. Not doing so results in exception errors when you attempt to create to-do tasks.

To switch from non-SSL to SSL configurations with Oracle BPM Worklist:

1. Log in to Oracle WebLogic Server Administration Console.
2. In the **Environment** section, select **Servers**.
3. Select the name of the managed server (for example, **soa\_server1**).
4. Select **Protocols**, then select **HTTP**.
5. In the **Frontend Host** field, enter the hostname on which Oracle BPM Worklist is located.
6. In the **Frontend HTTPS Port** field, enter the SSL listener port.
7. Click **Save**.

#### Configuring SOA Composite Applications for Two-Way SSL Communication

Oracle SOA Suite uses both Oracle WebLogic Server and Sun Secure Socket Layer (SSL) stacks for two-way SSL configurations.

* For the inbound web service bindings, Oracle SOA Suite uses the Oracle WebLogic Server infrastructure and, therefore, the Oracle WebLogic Server libraries for SSL.
* For the outbound web service bindings, Oracle SOA Suite uses JRF HttpClient and, therefore, the Sun JDK libraries for SSL.

Due to this difference, start Oracle WebLogic Server with the following JVM option.

To configure SOA composite applications for two-way SSL communication:

1. Open the following file:
   * On UNIX operating systems, open $MIDDLEWARE\_HOME/user\_projects/domains/*domain\_name*/bin/setDomainEnv.sh.
   * On Window operating systems, open MIDDLEWARE\_HOME\user\_projects\domains\*domain\_name*\bin\setDomainEnv.bat.
2. Add the following lines in the JAVA\_OPTIONS section, if the server is enabled for one-way SSL (server authorization only):

-Djavax.net.ssl.trustStore=*your\_truststore\_location*

For two-way SSL, the keystore information (location and password) is not required.

In addition, perform the following steps to enable two-way SSL for a SOA composite application to invoke another SOA composite application or another non-SOA application.

**Note:**

Both the server and client are assumed to have been configured for SSL with mutual authentication.

To enable two-way SSL for a SOA composite application to invoke another application:

1. On the client side, provide the keystore location.
   1. From the **SOA Infrastructure** menu, select **SOA Administration** > **Common Properties**.
   2. At the bottom of the page, click **More SOA Infra Advanced Configuration Properties**.
   3. Click **KeystoreLocation**.
   4. In the **Value** column, enter the keystore location.
   5. Click **Apply**.
   6. Click **Return**.
2. During design time in Oracle JDeveloper, update the reference section in the composite.xml file with the oracle.soa.two.way.ssl.enabled property.

<reference name="Service1"

ui:wsdlLocation=". . .">

<interface.wsdl interface=". . ."/>

<binding.ws port=". . .">

<property name="oracle.soa.two.way.ssl.enabled">true</property>

</binding.ws>

</reference>

1. In Oracle Enterprise Manager Fusion Middleware Control, select **WebLogic Domain** > ***domain\_name***.
2. Right-click ***domain\_name*** and select **Security** > **Credentials**.
3. Click **Create Map**.
4. In the **Map Name** field, enter a name (for example, SOA), and click **OK**.
5. Click **Create Key**.
6. Enter the following details.

| **Field** | **Description** |
| --- | --- |
| **Select Map** | Select the map created in Step [6](https://docs.oracle.com/cd/E29597_01/admin.1111/e10226/soacompapp_secure.htm#BABBAEAG) (for this example, **SOA**). |
| **Key** | Enter the key name (KeystorePassword is the default). |
| **Type** | Select **Password**. |
| **User Name** | Enter the keystore user name (KeystorePassword is the default). |
| **Password** | Enter the password that you created for the keystore. |

**Note:**

When you set up SSL in Oracle WebLogic Server, a key alias is required. You must enter mykey as the alias value. This value is required.

1. Set the keystore location in Oracle Enterprise Manager Fusion Middleware Control. See Step [1](https://docs.oracle.com/cd/E29597_01/admin.1111/e10226/soacompapp_secure.htm#CHDFCEHE) for instructions.
2. Modify composite.xml to use https and *sslport* to invoke a SOA composite application. For example, change the syntax shown in bold:

<?xml version="1.0" encoding="UTF-8" ?>

<!-- Generated by Oracle SOA Modeler version 1.0 at [4/1/09 11:01 PM]. -->

<composite name="InvokeEchoBPELSync"

revision="1.0"

label="2009-04-01\_23-01-53\_994"

mode="active"

state="on"

xmlns="http://xmlns.oracle.com/sca/1.0"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"

xmlns:orawsp="http://schemas.oracle.com/ws/2006/01/policy"

xmlns:ui="http://xmlns.oracle.com/soa/designer/">

<import

namespace="http://xmlns.oracle.com/CustomApps/InvokeEchoBPELSync/BPELProcess1"

location="BPELProcess1.wsdl" importType="wsdl"/>

<import namespace="http://xmlns.oracle.com/CustomApps/EchoBPELSync/

BPELProcess1"location="http://hostname:port/soa-infra/services/default/EchoBPEL

Sync/BPELProcess1.wsdl"

importType="wsdl"/>

to use https and *sslport*:

location="https://hostname:sslport/soa-infra/services/default/EchoBPELSync

/BPELProcess1.wsdl"

#### Invoking References in One-Way SSL Environments in Oracle JDeveloper

When invoking a web service as an external reference from a SOA composite application in one-way SSL environments, ensure that the certificate name (CN) and the hostname of the server exactly match. This ensures a correct SSL handshake.

For example, if a web service is named adfbc and the certificate has a server name of myhost05, the following syntax results in an SSL handshake exception.

<import namespace="/adfbc1/common/"

location="https://myhost05.us.oracle.com:8002/CustomApps-adfbc1-context-root/Ap

pModuleService?WSDL"

importType="wsdl"/>

<import namespace="/adfbc1/common/" location="Service1.wsdl"

importType="wsdl"/>

If you switch the order of import, the SSL handshake passes.

<import namespace="/adfbc1/common/" location="Service1.wsdl"

importType="wsdl"/>

<import namespace="/adfbc1/common/"

location="https://myhost05.us.oracle.com:8002/CustomApps-adfbc1-context-root/Ap

pModuleService?WSDL"

importType="wsdl"/>

Note the following restrictions around this issue:

* There are no options for ignoring hostname verification in Oracle JDeveloper as exist with the Oracle WebLogic Server Administration Console. This is because the SSL kit used by Oracle JDeveloper is different. Only the trust store can be configured from the command line. All other certificate arguments are not passed.
* In the WSDL file, https://*hostname* must match with that in the certificate, as described above. You cannot perform the same procedures as you can with a browser. For example, if the hostname is myhost05.us.oracle.com in the certificate's CN, then you can use myhost05, myhost05.us.oracle.com, or the IP address from a browser. In Oracle JDeveloper, always use the same name as in the certificate (that is, myhost05.us.oracle.com).

#### Configuring Oracle SOA Suite and Oracle HTTP Server for SSL Communication

Follow these steps to configure SSL communication between Oracle SOA Suite and Oracle HTTP Server.

#### Configuring Oracle HTTP Server for SSL Communication

To configure Oracle HTTP server for SSL communication:

1. Update mod\_ssl.conf with the <Location /integration/services> location directive.

LoadModule weblogic\_module ${ORACLE\_HOME}/ohs/modules/mod\_wl\_ohs.so

<IfModule mod\_weblogic.c>

WebLogicHost host.domain.com

WLLogFile <logdir>/ohs\_ssl.log

Debug ALL

DebugConfigInfo ON

SecureProxy ON

MatchExpression \*.jsp

WlSSLWallet <OHS\_

HOME>/instances/instance1/config/OHS/ohs1/keystores/default

</IfModule>

<Location /soa-infra>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /b2bconsole>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /b2b>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /integration/worklistapp>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /integration/services>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /DefaultToDoTaskFlow>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /OracleBAM>

WebLogicPort 9002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /OracleBAMWS>

WebLogicPort 9002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

<Location /sdpmessaging/userprefs-ui/>

WebLogicPort 8002

SetHandler weblogic-handler

ErrorPage http://host.domain.com:port/error.html

</Location>

1. Start the Oracle WebLogic Servers as described in [Section 5.6.4, "Configuring SOA Composite Applications for Two-Way SSL Communication."](https://docs.oracle.com/cd/E29597_01/admin.1111/e10226/soacompapp_secure.htm#CHDFJEFB)

#### Configuring Certificates for Oracle Client, Oracle HTTP Server, and Oracle WebLogic Server

To configure certificates for Oracle Client, Oracle HTTP Server, and Oracle WebLogic Server:

1. Export the user certificate from the Oracle HTTP Server wallet.

orapki wallet export -wallet . -cert cert.txt -dn 'CN=\"Self-Signed

Certificate for ohs1 \",OU=OAS,O=ORACLE,L=REDWOODSHORES,ST=CA,C=US'

1. Import the above certificate into the Oracle WebLogic Server trust store as a trusted certificate.

keytool -file cert.txt -importcert -trustcacerts -keystore DemoTrust.jks

1. Export the certificate from the Oracle WebLogic Server trust store.

keytool -keystore DemoTrust.jks -exportcert -alias wlscertgencab -rfc -file

certgencab.crt

1. Import the above certificate to the Oracle HTTP Server wallet as a trusted certificate.

orapki wallet add -wallet . -trusted\_cert -cert certgencab.crt -auto\_login\_only

1. Restart Oracle HTTP Server.
2. Restart the Oracle WebLogic Servers as described in [Section 5.6.4, "Configuring SOA Composite Applications for Two-Way SSL Communication."](https://docs.oracle.com/cd/E29597_01/admin.1111/e10226/soacompapp_secure.htm#CHDFJEFB)

#### Configuring SSL Between SOA Composite Application Instances and Oracle WebCache

The Test Web Service page in an Oracle WebCache and Oracle HTTP Server environment may require communication back through Oracle WebCache. Therefore, SSL must be configured between the SOA composite application instance and Oracle WebCache (that is, export the user certificate from the Oracle WebCache wallet and import it as a trusted certificate in the Oracle WebLogic Server trust store).

#### Using a Custom Trust Store for One-Way SSL During Design Time

To invoke a SOA composite application from another composite over HTTPS when using a custom trust store created with a tool such as keytool or orapki, perform the following actions in Oracle JDeveloper.

To use a custom trust store for one-way SSL during design time:

1. To fetch a WSDL file in the reference section, set the trust store information in **Tools** > **Preferences** > **Http Analyzer** > **HTTPS Setup** > **Client Trusted Certificate Keystore**.
2. During deployment to an SSL-enabled server, use the JSSE property at the command line:

jdev -J-Djavax.net.ssl.trustStore=*your\_trusted\_location*

#### Enabling an Asynchronous Process Deployed to an SSL-Enabled, Managed Server to Invoke Another Asynchronous Process Over HTTP

Assume you create the following environment:

* Asynchronous BPEL process A invokes asynchronous BPEL process B
* Asynchronous BPEL process A is deployed to a one-way SSL enabled, managed server
* All WSDL reference and bindings use plain HTTP

At runtime, the WSDL is looked for over HTTPS, and the callback message from asynchronous BPEL process B fails.

To resolve this issue, the callbackServerURL property must be passed at the reference binding level in the composite.xml file. This explicitly indicates the value of the callback URL for the given reference invocation. If the client composite is running in an SSL-managed server, then the callback defaults to SSL.

<reference name="Service1"

ui:wsdlLocation="http://localhost:8000/soa-infra/services/default/AsyncSecondB

PELMTOM/BPELProcess1.wsdl">

<interface.wsdl

interface="http://xmlns.oracle.com/Async/AsyncSecondBPELMTOM/BPELProcess1#wsdl

.interface(BPELProcess1)"

callbackInterface="http://xmlns.oracle.com/Async/AsyncSecondBPELMTOM/BPELProce

ss1#wsdl.interface(BPELProcess1Callback)"/>

<binding.ws

port="http://xmlns.oracle.com/Async/AsyncSecondBPELMTOM/BPELProcess1#wsdl.endp

oint(bpelprocess1\_client\_ep/BPELProcess1\_pt)"

location="http://localhost:8000/soa-infra/services/default/AsyncSecondBPELMTOM

/bpelprocess1\_client\_ep?WSDL">

<wsp:PolicyReference URI="oracle/wss\_username\_token\_client\_policy"

orawsp:category="security"

orawsp:status="enabled"/>

<wsp:PolicyReference URI="oracle/wsaddr\_policy"

orawsp:category="addressing"

orawsp:status="enabled"/>

.

<property name="callbackServerURL">http://localhost:8000/</property>

.

</binding.ws>

.

<callback>

<binding.ws

port="http://xmlns.oracle.com/Async/AsyncSecondBPELMTOM/BPELProcess1#wsdl.endp

oint(bpelprocess1\_client\_ep/BPELProcess1Callback\_pt)">

<wsp:PolicyReference

URI="oracle/wss\_username\_token\_service\_policy"

orawsp:category="security"

orawsp:status="enabled"/>

</binding.ws>

</callback>

.

</reference>