RMI / IIOP Interoperability
A Liberty Server for z /OS
Sample with Security

Keith Jabcuga kjabcuga@us.ibm.com
Justin McNeil jdmcneil@us.ibm.com
WebSphere Software Support for z/OS
Poughkeepsie, NY
February 28, 2019

Introduction

The InteropV9 enterprise application EAR file is a JEE application that performs an RMI/IIOP call from one platform to another under different security scenarios. This can be useful for diagnosing setup or run-time problems, and for understanding how to configure Liberty to perform remote invocations across platforms. The scenarios covered in this paper involve the InteropV9 application deployed on a Liberty application server on Windows platform calling another instance of the InteropV9 application deployed on a Liberty application server running on the z/OS platform. The InteropV9 application includes a servlet calling a remote stateless session bean. In each of the scenarios, the CSIv2 security settings are modified to illustrate different approaches to performing a secure remote EJB call.

The scenarios covered in this document include:

No Security
- No SSL and No Authentication

CSIv2 Transport Layer
- Securing the transport with SSL

CSIv2 Attribute Layer / CSIv2 Authentication Layer
- Identity Assertion – Trust with LTPA
- Identity Assertion – Trust with Basic Authentication (GSSUP)

CSIv2 Attribute Layer / CSIv2 Transport Layer
- Identity Assertion – Client Certificate Authentication

Common Scenarios and Outcomes

The following table summarizes the scenarios covered in this document, and the configuration settings needed to ensure the identity is passed to from the client Liberty Windows server to the remote Liberty for z/OS server.

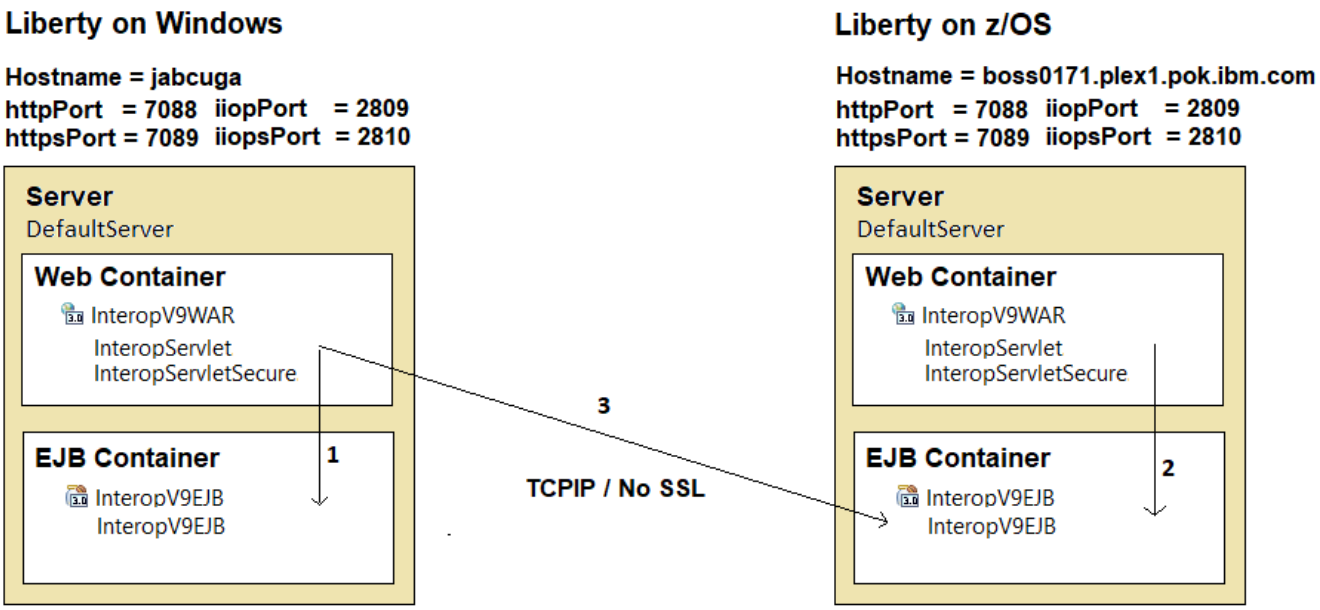
Liberty Windows and Liberty z/OS common settings						
Liberty Windows	**CSIv2 Transport Layer**		**CSIv2 Authentication Layer**		**CSIv2 Attribute Layer**	**Liberty z/OS**
Client Identity	**sslEnabled**	**ClientAuthentication**	**establishTrustInClient**	**mechanisms**	**IdentityAssertion**	**Server Identity**
Unauthenticated	true	false	Never	Not Applicable	false	Unauthenticated
Userid	true	false	Never	Not Applicable	false	Userid
Unauthenticated	true	false	Required	LTPA	true	Unauthenticated
Userid	true	false	Required	LTPA	true	Userid
Unauthenticated	true	false	Required	GSSUP	true	Unauthenticated
Userid	true	false	Required	GSSUP	true	Userid
Unauthenticated	true	true	Never	Not Applicable	true	Unauthenticated
Userid	true	true	Never	Not Applicable	true	Userid

The following table illustrates some of misconfigured settings and error or identity that resulted on the remote Liberty for z/OS server.

Possible misconfiguration or unexpected outcome						
Liberty Windows	**CSIv2 Transport Layer**		**CSIv2 Authentication Layer**		**CSIv2 Attribute Layer**	**Liberty z/OS**
Client Identity	**sslEnabled**	**ClientAuthentication**	**establishTrustInClient**	**mechanisms**	**IdentityAssertion**	**Server Identity**
Unauthenticated	true	false	Required	LTPA	IdentityAssertion=false	No_Permission Failure
Userid	true	false	Required	LTPA	IdentityAssertion=false	Userid
Unauthenticated	true	false	Required	GSSUP	IdentityAssertion=false	Failure/No GSSUP token
Userid	true	false	Required	GSSUP	IdentityAssertion=false	Failure/No GSSUP token
Unauthenticated	true	false	Never	Not Applicable	IdentityAssertion=false	Certificate ID (ie. cn=USERID)
Userid	true	false	Never	Not Applicable	IdentityAssertion=false	Userid
Unauthenticated	true	true	Required	No Mechanism	IdentityAssertion=false	Failure No LTPA/GSSUP token
Userid	true	true	Required	No Mechanism	IdentityAssertion=false	Failure No LTPA/GSSUP token

No Security

When trying to install and run the InteropV9 application, its easier to diagnose any issues with the network if you test initially with security off. For example, hostname DNS resolution, multiple TCPIP stacks, reserved ports, firewalls, and any other type of network configuration that might prevent the communication from one platform to another.



1. The InteropEARV9 application can be run on within the same Windows server in which the InteropV9Servlet makes an RMI-IIOP call over the network to the InteropV9EJB running in the same Windows server.

2. The InteropEARV9 application can be run on within the same z/OS server in which the InteropV9Servlet makes an RMI-IIOP call over the network to the InteropV9EJB running in the same z/OS server.

3. The InteropEARV9 application can be run on on the Windows server in which the InteropV9Servlet makes an RMI-IIOP call over the network to the InteropV9EJB running in the z/OS server.

server.xml – Liberty on Windows and z/OS

For initial testing, the InteropV9.ear file was placed in the same directory as the server.xml file and is referenced using a relative path with variable ${server.config.dir}. A minimal set of features are enabled with appSecurity-3.0,ssl-1.0,and transportSecurity-1.0 removed in order to disable security and SSL in the server. The only changes that are needed in the two server.xml files are the hostname and port for the Windows and z/OS Liberty servers.

The Liberty server running on Windows is running with hostname “jabcuga” on port 2809.

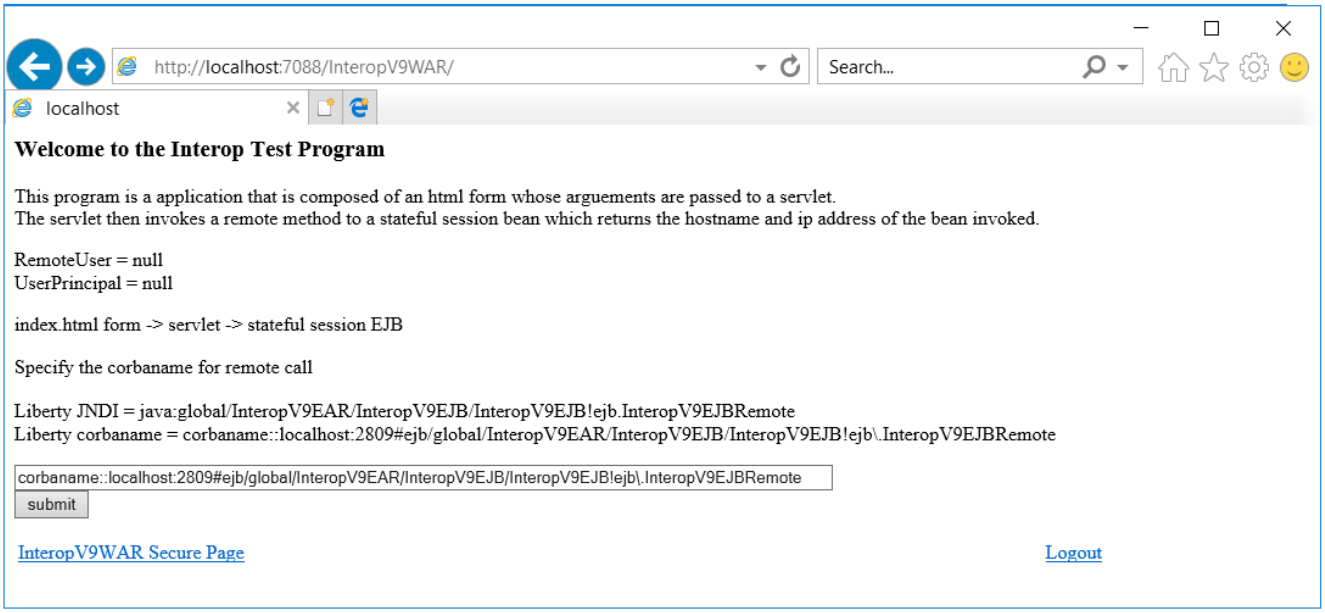
```
<server description="LibertyWindows">

   <featureManager>
      <feature>jsp-2.3</feature>
      <feature>servlet-4.0</feature>
      <feature>ejb-3.2</feature>
   </featureManager>

   <httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
   </httpEndpoint>

   <iiopEndpoint id="defaultIiopEndpoint" iiopPort="2809" host="jabcuga">
   </iiopEndpoint>

  <enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  </enterpriseApplication>

</server>
```

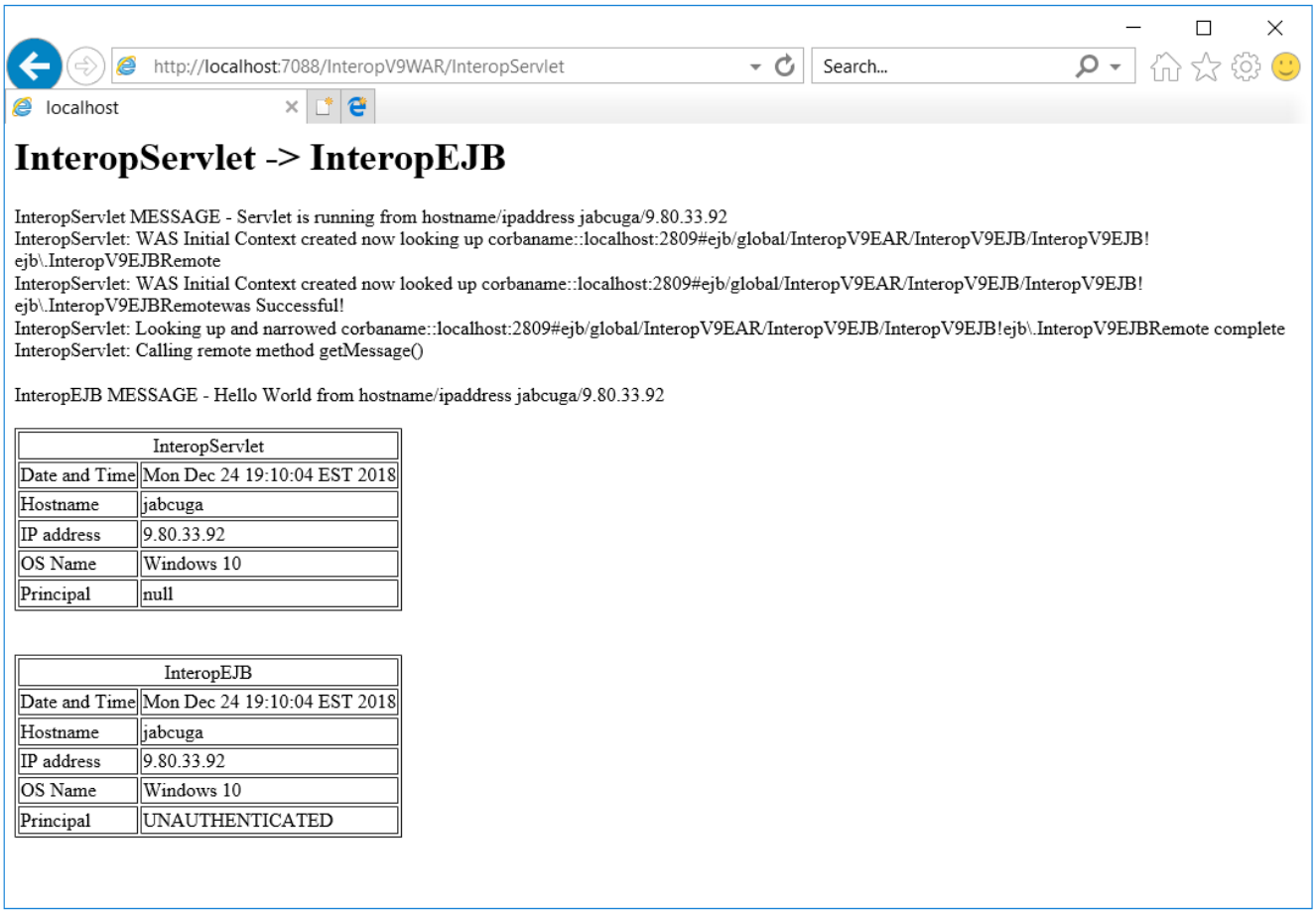
The Liberty server running on z/OS is running with hostname “boss0181.pok.ibm.com” on port 2809.

```
<server description="LibertyZOS">

   <featureManager>
      <feature>jsp-2.3</feature>
      <feature>servlet-4.0</feature>
      <feature>ejb-3.2</feature>
   </featureManager>

   <httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
   </httpEndpoint>

   <iiopEndpoint id="defaultIiopEndpoint" iiopPort="2809" host="boss0181.pok.ibm.com">
   </iiopEndpoint>

  <enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  </enterpriseApplication>

</server>
```

Windows to Windows

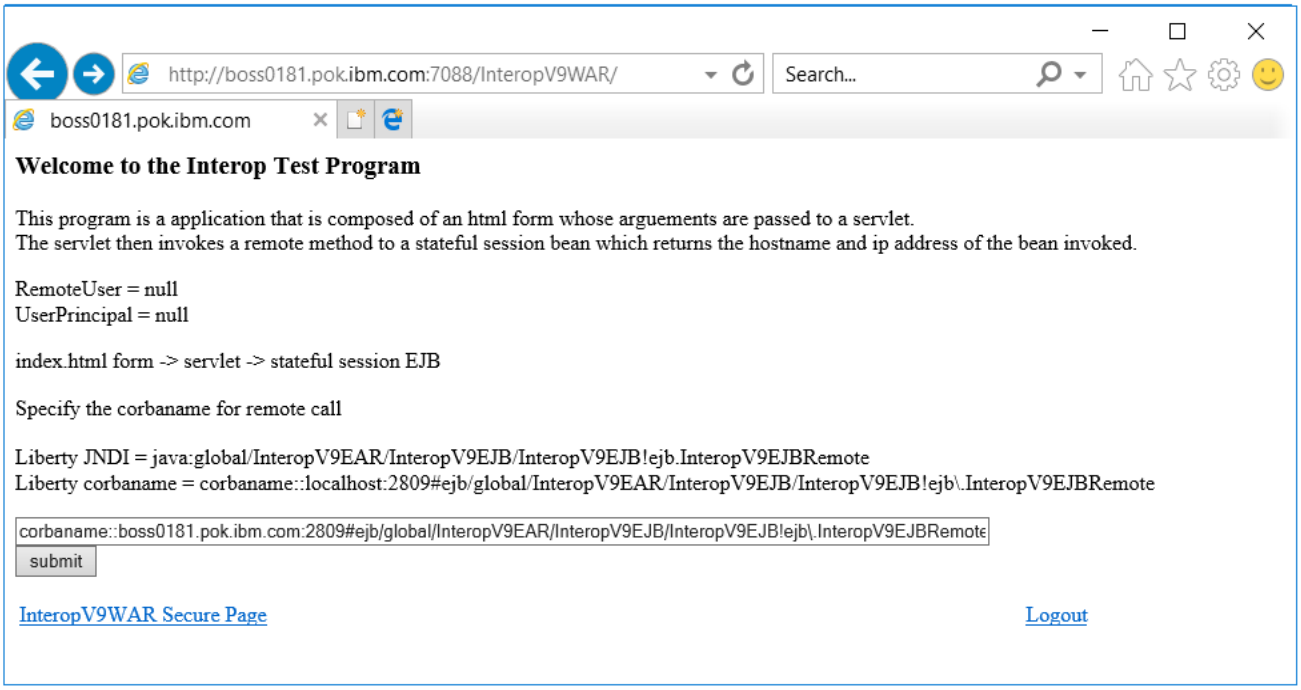
The following images show how the remote call where the client InteropServlet calls the InteropEJB running in the same Liberty for Windows Server.

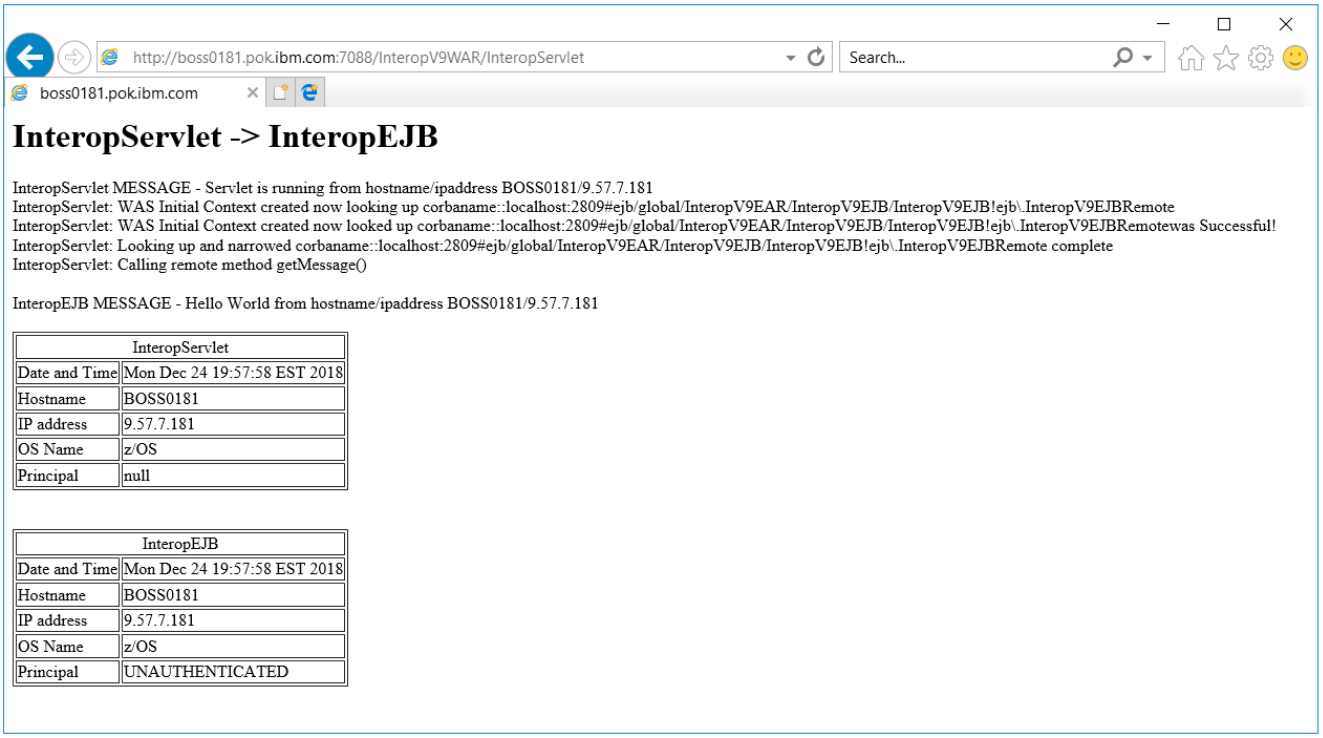




z/OS to z/OS

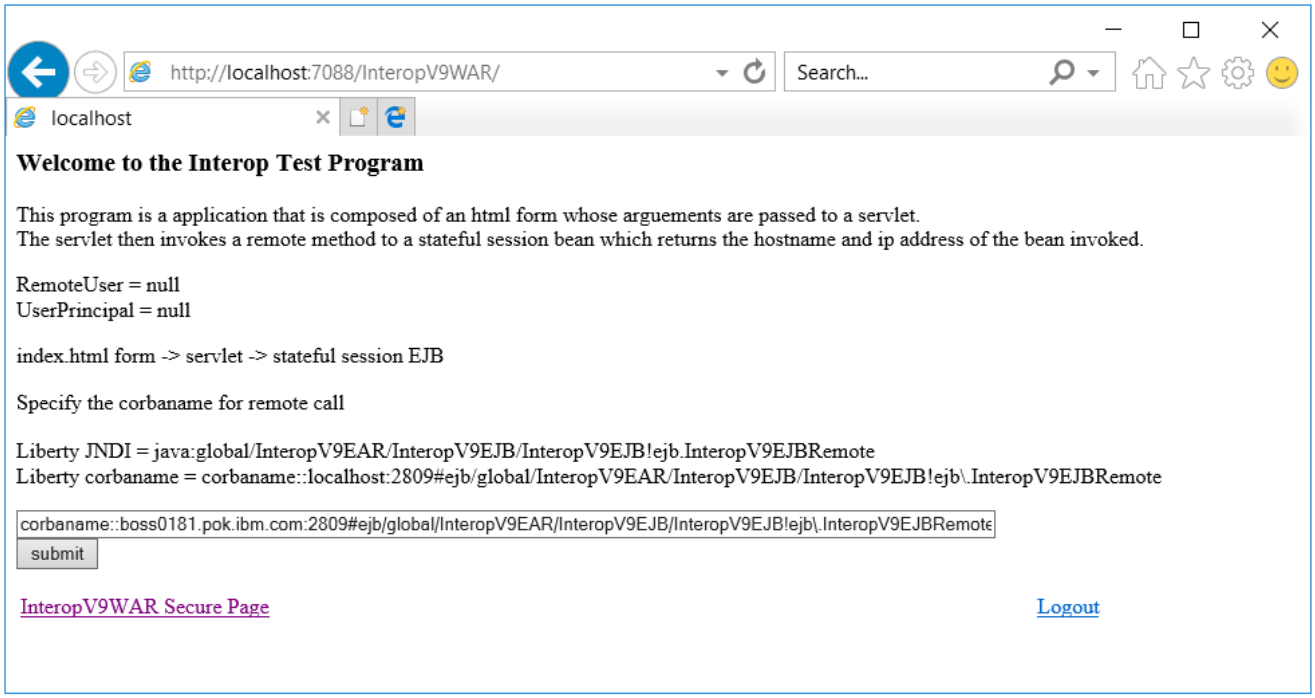
The following images show how the remote call where the client InteropServlet calls the InteropEJB running in the same Liberty for z/OS Server.

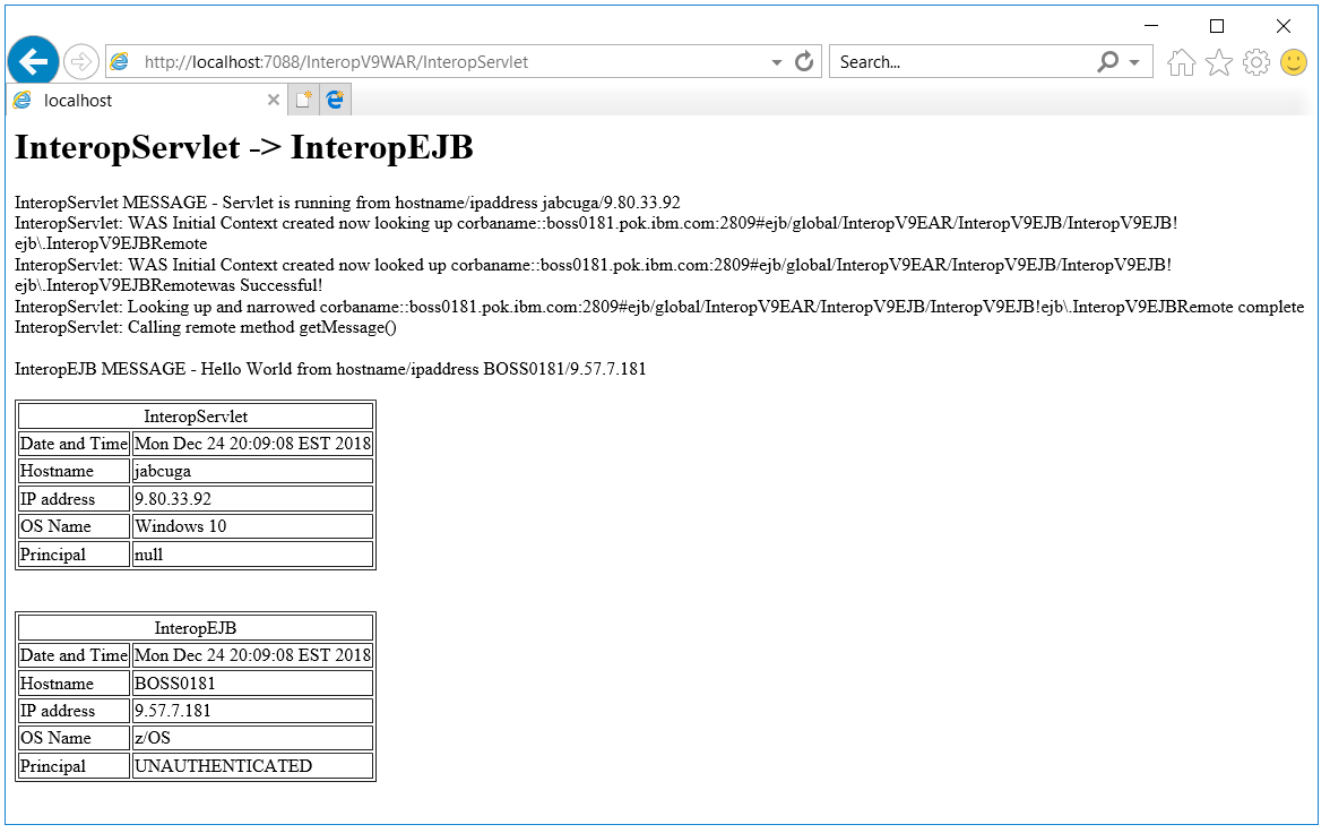




Windows to z/OS

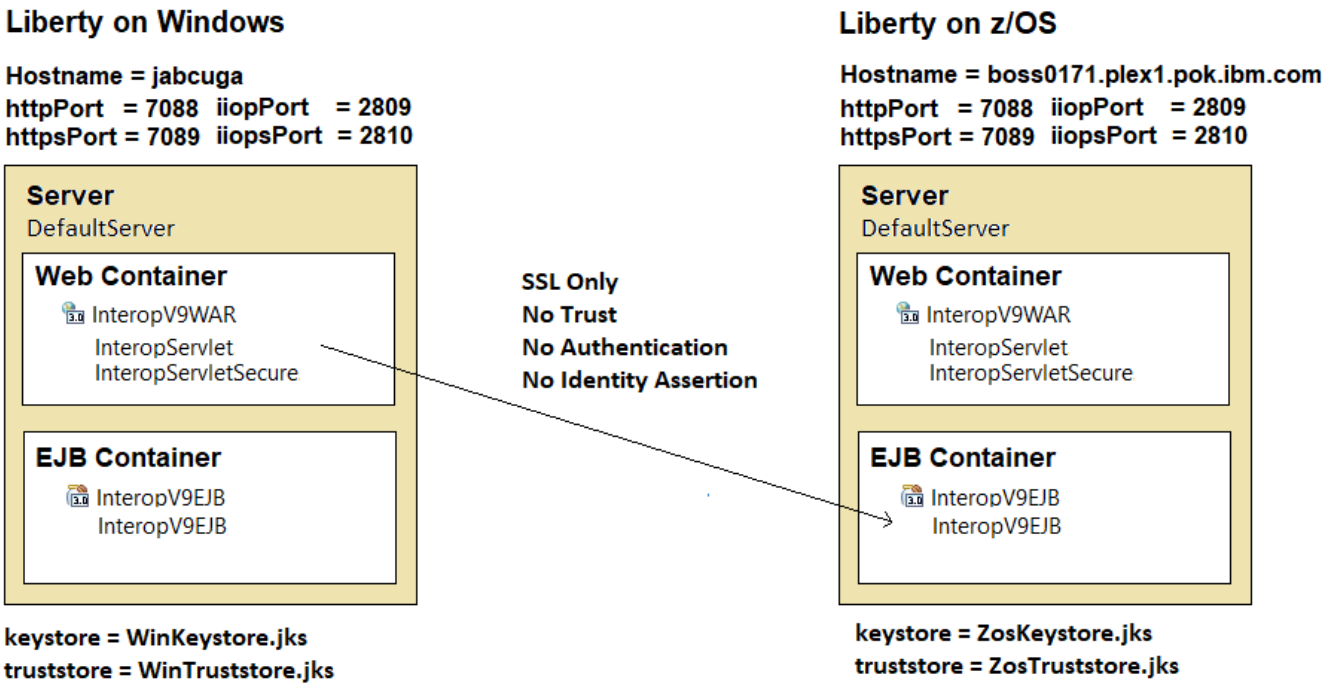
The following images show how the remote call where the client InteropServlet running on Liberty for Windows calls the InteropEJB running in the same Liberty for z/OS Server.





CSIv2 Transport Layer - Securing the transport with SSL

Once the InteropV9 application has been tested cross platform successfully with security disabled the next step would be enable SSL to secure the transport. In order to focus on SSL only the serverPolicy.csiv2 and clientPolicy.csiv2 tags can be added to the server.xml to disable establishing trust between the two Liberty servers and to disable Identity Assertion.



The InteropEARV9 application can be run on on the Windows server in which the InteropV9Servlet makes an SSL call over the network to the InteropV9EJB running in the z/OS server. This step would require exchanging signer certificates between Liberty for z/OS and Liberty on Windows.

The next few sections offer 2 ways to create the certificates.

- “Using self-signed certificates” which uses the keytool command provided by Java
- “Using a personal certificate Signed by a Certificate Authority” which makes use of the openssl utility to create the certificates and keytool to import them into the keystore and truststore.

The section “Using self-signed certificates” can be used in a test environment to go through the SSL setup and configuration in Liberty. “Using a personal certificate Signed by a Certificate Authority” section would more closely resemble a personal certificate and signer certificate that a user might receive from a trusted certificate authority. The steps in only one of these sections would be needed to test the SSL setup.

Using self-signed certificates

Its possible to create a self signed certificate by adding the keystore tag to server.xml

```
<keyStore id="defaultKeyStore" password="Liberty" />
```

or with the Liberty command line option

```
securityUtility createSSLCertificate
```

However, in this setup will use the keytool command provided by the JVM in order to facilitate creating certificates, keystore and truststore files as well as exporting and importing the certificates into the keystore and truststore files.

Keytool commands to create self- signed certificates

Generate 2 self signed certificates in keystore files WinKeystore.jks and ZosKeystore.jks

```
keytool -genkeypair -dname "CN=jabcuga" -alias WinSelfSigned -keyalg RSA -keysize 2048
-keystore WinKeystore.jks -validity 365 -storetype JKS -storepass Liberty -keypass Liberty

keytool -genkeypair -dname "CN=boss0181.pok.ibm.com" -alias ZosSelfSigned -keyalg RSA -keysize 2048
-keystore ZosKeystore.jks -validity 365 -storetype JKS -storepass Liberty -keypass Liberty
```

Export the 2 self signed certificates from the keystore to certificate files WinSelfSigned.crt and ZosSelfSigned.crt

```
keytool -exportcert -alias WinSelfSigned -file WinSelfSigned.crt -keystore WinKeystore.jks -storepass Liberty
keytool -exportcert -alias ZosSelfSigned -file ZosSelfSigned.crt -keystore ZosKeystore.jks -storepass Liberty
```

Print the details of the certificates in the 2 certificate files

```
keytool -printcert -v -file WinSelfSigned.crt
keytool -printcert -v -file ZosSelfSigned.crt
```

Import the Windows and ZOS certificates into the Windows truststore file WinTruststore.jks

```
keytool -import -trustcacerts -alias WinSelfSigned -file WinSelfSigned.crt -keystore WinTruststore.jks -storepass Liberty
-noprompt

keytool -import -trustcacerts -alias ZosSelfSigned -file ZosSelfSigned.crt -keystore WinTruststore.jks -storepass Liberty
-noprompt
```

Import the Windows and ZOS certificates into the ZOS truststore file ZosTruststore.jks

```
keytool -import -trustcacerts -alias WinSelfSigned -file WinSelfSigned.crt -keystore ZosTruststore.jks -storepass Liberty
-noprompt

keytool -import -trustcacerts -alias ZosSelfSigned -file ZosSelfSigned.crt -keystore ZosTruststore.jks -storepass Liberty
-noprompt
```

Check the contents of the Windows keystore and truststore

```
keytool -list -v -keystore WinKeystore.jks -storepass Liberty
keytool -list -v -keystore WinTruststore.jks -storepass Liberty
```

Check the contents of the Zos keystore and trustsore

```
keytool -list -v -keystore ZosKeystore.jks -storepass Liberty
keytool -list -v -keystore ZosTruststore.jks -storepass Liberty
```

Windows keystore and truststore

The commands below show the contents of the Windows keystore and truststore files.

keytool -list -v -keystore WinKeystore.jks -storepass Liberty

Your keystore contains 1 entry

Alias name: **winselfsigned**
Creation date: Jan 6, 2019
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=jabcuga
Issuer: CN=jabcuga
Serial number: 7b85fbb6
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: 8D:96:A7:5C:FC:C9:6F:B9:18:F1:A6:96:3F:C1:18:DC
SHA1: 7E:87:93:E9:C9:EC:E9:32:61:B4:E1:C5:D1:11:83:7B:CD:A5:D0:5C
SHA256: 7A:96:F6:F3:B8:51:57:57:69:A5:85:35:56:5A:FF:17:BC:78:9D:DB:FE:61:1B:5A:1F:58:19:B0:8E:66:EE:E3
Signature algorithm name: SHA256withRSA
Version: 3

keytool -list -v -keystore WinTruststore.jks -storepass Liberty

Your keystore contains 2 entries

Alias name: **winselfsigned**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=jabcuga
Issuer: CN=jabcuga
Serial number: 7b85fbb6
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: 8D:96:A7:5C:FC:C9:6F:B9:18:F1:A6:96:3F:C1:18:DC
SHA1: 7E:87:93:E9:C9:EC:E9:32:61:B4:E1:C5:D1:11:83:7B:CD:A5:D0:5C
SHA256: 7A:96:F6:F3:B8:51:57:57:69:A5:85:35:56:5A:FF:17:BC:78:9D:DB:FE:61:1B:5A:1F:58:19:B0:8E:66:EE:E3
Signature algorithm name: SHA256withRSA
Version: 3

Alias name: **zosselfsigned**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=boss0181.pok.ibm.com
Issuer: CN=boss0181.pok.ibm.com
Serial number: 5497766a
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: EB:F2:8A:9C:8D:D8:66:4A:07:5C:4C:D2:BD:87:04:61
SHA1: 24:0D:6F:60:30:41:7A:C5:57:18:69:DE:6C:04:56:EB:15:8B:D0:C8
SHA256: D3:D6:2B:B7:CC:A2:56:30:C8:B0:A4:AF:A4:2A:17:90:F3:73:F8:2D:D1:9E:60:BF:3A:9F:C5:AB:5B:7F:D4:93
Signature algorithm name: SHA256withRSA
Version: 3

z/OS keystore and truststore

The commands below show the contents of the z/OS keystore and truststore files.

keytool -list -v -keystore ZosKeystore.jks -storepass Liberty

Your keystore contains 1 entry

Alias name: **zosselfsigned**
Creation date: Jan 6, 2019
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=boss0181.pok.ibm.com
Issuer: CN=boss0181.pok.ibm.com
Serial number: 5497766a
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: EB:F2:8A:9C:8D:D8:66:4A:07:5C:4C:D2:BD:87:04:61
SHA1: 24:0D:6F:60:30:41:7A:C5:57:18:69:DE:6C:04:56:EB:15:8B:D0:C8
SHA256: D3:D6:2B:B7:CC:A2:56:30:C8:B0:A4:AF:A4:2A:17:90:F3:73:F8:2D:D1:9E:60:BF:3A:9F:C5:AB:5B:7F:D4:93
Signature algorithm name: SHA256withRSA
Version: 3

keytool -list -v -keystore ZosTruststore.jks -storepass Liberty

Your keystore contains 2 entries

Alias name: **winselfsigned**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=jabcuga
Issuer: CN=jabcuga
Serial number: 7b85fbb6
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: 8D:96:A7:5C:FC:C9:6F:B9:18:F1:A6:96:3F:C1:18:DC
SHA1: 7E:87:93:E9:C9:EC:E9:32:61:B4:E1:C5:D1:11:83:7B:CD:A5:D0:5C
SHA256: 7A:96:F6:F3:B8:51:57:57:69:A5:85:35:56:5A:FF:17:BC:78:9D:DB:FE:61:1B:5A:1F:58:19:B0:8E:66:EE:E3
Signature algorithm name: SHA256withRSA
Version: 3

Alias name: **zosselfsigned**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=boss0181.pok.ibm.com
Issuer: CN=boss0181.pok.ibm.com
Serial number: 5497766a
Valid from: 1/6/19 9:09 AM until: 1/6/20 9:09 AM
Certificate fingerprints:
MD5: EB:F2:8A:9C:8D:D8:66:4A:07:5C:4C:D2:BD:87:04:61
SHA1: 24:0D:6F:60:30:41:7A:C5:57:18:69:DE:6C:04:56:EB:15:8B:D0:C8
SHA256: D3:D6:2B:B7:CC:A2:56:30:C8:B0:A4:AF:A4:2A:17:90:F3:73:F8:2D:D1:9E:60:BF:3A:9F:C5:AB:5B:7F:D4:93
Signature algorithm name: SHA256withRSA
Version: 3

Using a Personal Certificate Signed by a Certificate Authority

In this scenario, the openssl command is used to create a root CA signing certificate that signs a personal certificate. The openssl commands can be skipped if you already have a personal certificate and signer certificate from a certificate authority.

OpenSSL commands to create a Signer and Personal certificate

Generate a root signer CA Certificate

Generate private Key
openssl genrsa -out WinRootSigner.key 2048
openssl genrsa -out ZosRootSigner.key 2048

Generate CA Certificate Signing Request (CSR)
openssl req -new -sha256 -key WinRootSigner.key -out WinRootSigner.csr -subj "/CN=WinRootSigner"
openssl req -new -sha256 -key ZosRootSigner.key -out ZosRootSigner.csr -subj "/CN=ZosRootSigner"

Generate CA Certificate
openssl x509 -signkey WinRootSigner.key -in WinRootSigner.csr -req -days 3650 -out WinRootSigner.pem
openssl x509 -signkey ZosRootSigner.key -in ZosRootSigner.csr -req -days 3650 -out ZosRootSigner.pem

Generate Server Certificate signed by the CA Certificate

Generate private Key
openssl genrsa -out WinPersonal.key 2048
openssl genrsa -out ZosPersonal.key 2048

Generate Personal Certificate Signing Request (CSR)
openssl req -new -sha256 -key WinPersonal.key -out WinPersonal.csr -subj "/CN=jabcuga"
openssl req -new -sha256 -key ZosPersonal.key -out ZosPersonal.csr -subj "/CN=boss0181.pok.ibm.com"

Generate Personal Certificate
openssl x509 -req -in WinPersonal.csr -CA WinRootSigner.pem -CAkey WinRootSigner.key -CAcreateserial -out WinPersonal.crt -days 3650 -sha256

openssl x509 -req -in ZosPersonal.csr -CA ZosRootSigner.pem -CAkey ZosRootSigner.key -CAcreateserial -out ZosPersonal.crt -days 3650 -sha256

View the certificates
openssl x509 -text -noout -in WinRootSigner.pem
openssl x509 -text -noout -in WinPersonal.crt
openssl x509 -text -noout -in ZosRootSigner.pem
openssl x509 -text -noout -in ZosPersonal.crt

Convert the CA Signer PEM certificates to DER format
openssl x509 -outform der -in WinRootSigner.pem -out WinRootSigner.der
openssl x509 -outform der -in ZosRootSigner.pem -out ZosRootSigner.der

Convert Personal PEM certificate and a private key to password protected PKCS#12 (.p12) format
openssl pkcs12 -export -out WinPersonal.p12 -inkey WinPersonal.key -in WinPersonal.crt -certfile WinRootSigner.pem -passout pass:Liberty -alias “WinPersonal”

openssl pkcs12 -export -out ZosPersonal.p12 -inkey ZosPersonal.key -in ZosPersonal.crt -certfile ZosRootSigner.pem -passout pass:Liberty -alias “ZosPersonal”

Keytool Commands to import the Signer and Personal Certificate

The keytool command is used to import the root signing certificate and personal certificate created with the openssl commands into the corresponding keystore and truststore.

Import the personal certificate and private key from a password protected pkcs12 file into keystore

keytool -importkeystore -deststorepass Liberty -destkeystore WinKeystore.jks -srcstorepass Liberty -srckeystore WinPersonal.p12 -srcstoretype PKCS12 -alias WinPersonal

keytool -importkeystore -deststorepass Liberty -destkeystore ZosKeystore.jks -srcstorepass Liberty -srckeystore ZosPersonal.p12 -srcstoretype PKCS12 -alias ZosPersonal

Import the Root Signer certificate DER file into truststore

keytool -import -trustcacerts -alias WinRootSigner -file WinRootSigner.der -keystore WinTruststore.jks -storepass Liberty -noprompt

keytool -import -trustcacerts -alias ZosRootSigner -file ZosRootSigner.der -keystore WinTruststore.jks -storepass Liberty -noprompt

keytool -import -trustcacerts -alias ZosRootSigner -file ZosRootSigner.der -keystore ZosTruststore.jks -storepass Liberty -noprompt

keytool -import -trustcacerts -alias WinRootSigner -file WinRootSigner.der -keystore ZosTruststore.jks -storepass Liberty -noprompt

Check the contents of the Windows keystore and truststore

keytool -list -v -keystore WinKeystore.jks -storepass Liberty
keytool -list -v -keystore WinTruststore.jks -storepass Liberty

Check the contents of the Zos keystore and trustsore

keytool -list -v -keystore ZosKeystore.jks -storepass Liberty
keytool -list -v -keystore ZosTruststore.jks -storepass Liberty

Windows keystore and truststore

The commands below show the contents of the Windows keystore and truststore files.

keytool -list -v -keystore WinKeystore.jks -storepass Liberty

```
Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 1 entry

Alias name: WinPersonal
Creation date: Jan 6, 2019
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=jabcuga
Issuer: CN=WinRootSigner
Serial number: b78e5287624fbf93
Valid from: 1/6/19 1:41 PM until: 1/3/29 1:41 PM
Certificate fingerprints:
        MD5:  57:3F:58:13:45:58:B8:2E:48:DE:C3:88:CE:75:15:F0
        SHA1: 04:EF:4B:97:68:F3:13:BE:95:BC:32:BB:97:EF:F3:2B:03:EF:97:C8
        SHA256: 8E:8C:B3:E1:4E:42:61:CB:7D:82:C4:EF:82:C5:49:5D:9A:53:F7:04:4D:15:53:E9:99:E4:7F:C3:CE:48:2D:11
        Signature algorithm name: SHA256withRSA
        Version: 1
Certificate[2]:
Owner: CN=WinRootSigner
Issuer: CN=WinRootSigner
Serial number: cb55c54e1c7be635
Valid from: 1/6/19 1:40 PM until: 1/3/29 1:40 PM
Certificate fingerprints:
        MD5:  57:76:D3:68:D7:22:3C:54:B9:46:6B:5B:EC:95:15:D8
        SHA1: C3:63:A0:5E:CD:7D:85:5A:17:F8:3F:E7:79:62:EB:DC:3D:BF:AB:1D
        SHA256: 41:B5:07:FC:8E:1A:6F:BB:40:87:79:CE:06:64:B4:F6:FE:01:BF:01:54:A2:11:B5:C8:C7:52:06:51:53:D9:8E
        Signature algorithm name: SHA256withRSA
        Version: 1
```

keytool -list -v -keystore WinTruststore.jks -storepass Liberty

```
Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

Alias name: zosrootsigner
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=ZosRootSigner
Issuer: CN=ZosRootSigner
Serial number: af2f552a43d00e2a
Valid from: 1/6/19 2:19 PM until: 1/3/29 2:19 PM
Certificate fingerprints:
        MD5:  D1:22:41:BB:2A:A2:DC:81:1A:27:7A:57:5F:DD:57:9A
        SHA1: 84:4F:7A:3A:E7:BA:53:7D:F6:B8:66:FB:AE:FA:C5:90:F3:81:57:05
        SHA256: F4:1F:17:24:29:8A:45:35:22:E3:8F:83:74:C9:23:38:A4:5B:09:58:7E:22:7A:23:D9:DF:A0:3D:2C:88:C5:58
        Signature algorithm name: SHA256withRSA
        Version: 1
```

Alias name: **winrootsigner**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=**WinRootSigner**
Issuer: CN=**WinRootSigner**
Serial number: cb55c54e1c7be635
Valid from: 1/6/19 1:40 PM until: 1/3/29 1:40 PM
Certificate fingerprints:
MD5: 57:76:D3:68:D7:22:3C:54:B9:46:6B:5B:EC:95:15:D8
SHA1: C3:63:A0:5E:CD:7D:85:5A:17:F8:3F:E7:79:62:EB:DC:3D:BF:AB:1D
SHA256: 41:B5:07:FC:8E:1A:6F:BB:40:87:79:CE:06:64:B4:F6:FE:01:BF:01:54:A2:11:B5:C8:C7:52:06:51:53:D9:8E
Signature algorithm name: SHA256withRSA
Version: 1

z/OS keystore and truststore

The commands below show the contents of the z/OS keystore and truststore files.

keytool -list -v -keystore ZosKeystore.jks -storepass Liberty

```
Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 1 entry

Alias name: ZosPersonal
Creation date: Jan 6, 2019
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=boss0181.pok.ibm.com
Issuer: CN=ZosRootSigner
Serial number: 805635ab94ccaed9
Valid from: 1/6/19 2:20 PM until: 1/3/29 2:20 PM
Certificate fingerprints:
        MD5:  26:24:ED:FA:69:74:0E:1F:17:8D:78:97:57:2E:A0:78
        SHA1: E5:FA:BB:88:43:D8:8A:F7:3A:54:A8:B0:18:1C:C1:C6:1A:A1:4B:34
        SHA256: DB:F7:63:88:0D:37:83:7D:68:5F:8A:5F:5C:3A:00:86:04:15:0C:27:B0:71:75:CB:95:D7:76:72:91:9      7:73:4E
        Signature algorithm name: SHA256withRSA
        Version: 1
Certificate[2]:
Owner: CN=ZosRootSigner
Issuer: CN=ZosRootSigner
Serial number: af2f552a43d00e2a
Valid from: 1/6/19 2:19 PM until: 1/3/29 2:19 PM
Certificate fingerprints:
        MD5:  D1:22:41:BB:2A:A2:DC:81:1A:27:7A:57:5F:DD:57:9A
        SHA1: 84:4F:7A:3A:E7:BA:53:7D:F6:B8:66:FB:AE:FA:C5:90:F3:81:57:05
        SHA256: F4:1F:17:24:29:8A:45:35:22:E3:8F:83:74:C9:23:38:A4:5B:09:58:7E:22:7A:23:D9:DF:A0:3D:2C:8      8:C5:58
        Signature algorithm name: SHA256withRSA
        Version: 1
```

keytool -list -v -keystore ZosTruststore.jks -storepass Liberty

```
Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

Alias name: zosrootsigner
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=ZosRootSigner
Issuer: CN=ZosRootSigner
Serial number: af2f552a43d00e2a
Valid from: 1/6/19 2:19 PM until: 1/3/29 2:19 PM
Certificate fingerprints:
        MD5:  D1:22:41:BB:2A:A2:DC:81:1A:27:7A:57:5F:DD:57:9A
        SHA1: 84:4F:7A:3A:E7:BA:53:7D:F6:B8:66:FB:AE:FA:C5:90:F3:81:57:05
        SHA256: F4:1F:17:24:29:8A:45:35:22:E3:8F:83:74:C9:23:38:A4:5B:09:58:7E:22:7A:23:D9:DF:A0:3D:2C:88:C5:58
        Signature algorithm name: SHA256withRSA
        Version: 1
```

Alias name: **winrootsigner**
Creation date: Jan 6, 2019
Entry type: trustedCertEntry

Owner: CN=**WinRootSigner**
Issuer: CN=**WinRootSigner**
Serial number: cb55c54e1c7be635
Valid from: 1/6/19 1:40 PM until: 1/3/29 1:40 PM
Certificate fingerprints:
MD5: 57:76:D3:68:D7:22:3C:54:B9:46:6B:5B:EC:95:15:D8
SHA1: C3:63:A0:5E:CD:7D:85:5A:17:F8:3F:E7:79:62:EB:DC:3D:BF:AB:1D
SHA256: 41:B5:07:FC:8E:1A:6F:BB:40:87:79:CE:06:64:B4:F6:FE:01:BF:01:54:A2:11:B5:C8:C7:52:06:51:53:D9:8E
Signature algorithm name: SHA256withRSA
Version: 1

server.xml – Liberty on Windows

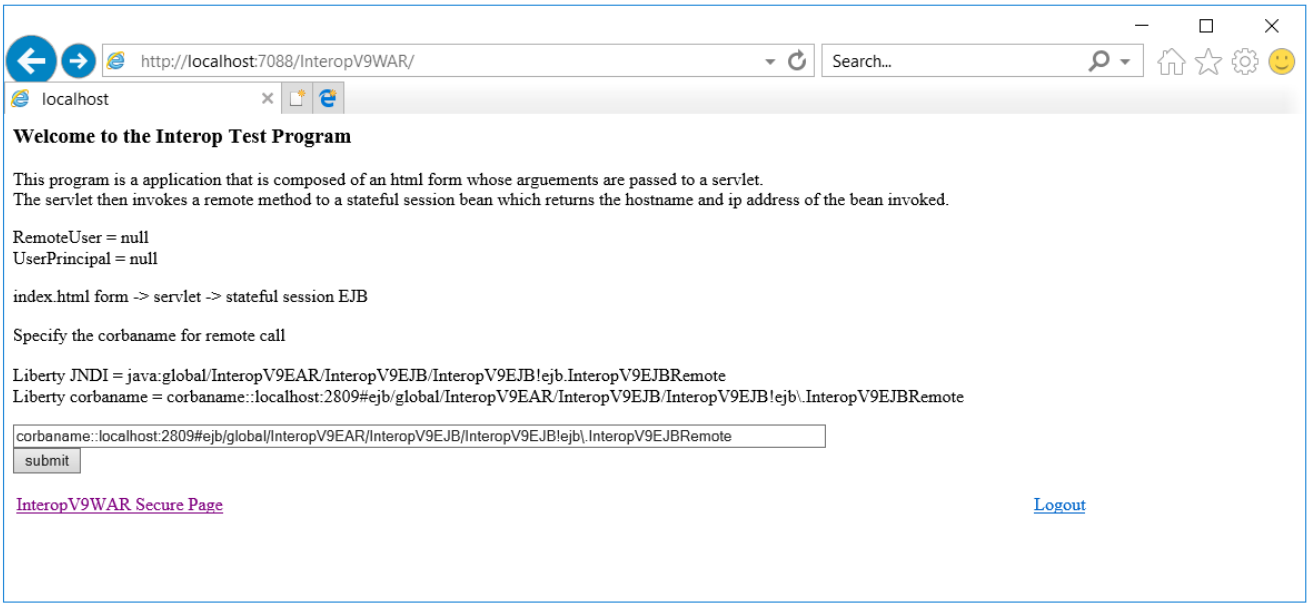
```
<server description="LibertyWindows">

<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="jabcuga" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
            clientAuthentication="false" clientAuthenticationSupported="false" />

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/WinKeystore.jks"
            password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/WinTruststore.jks"
            password="Liberty" type="JKS" />

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never"/>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>

<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>

</server>
```

server.xml – Liberty on z/OS

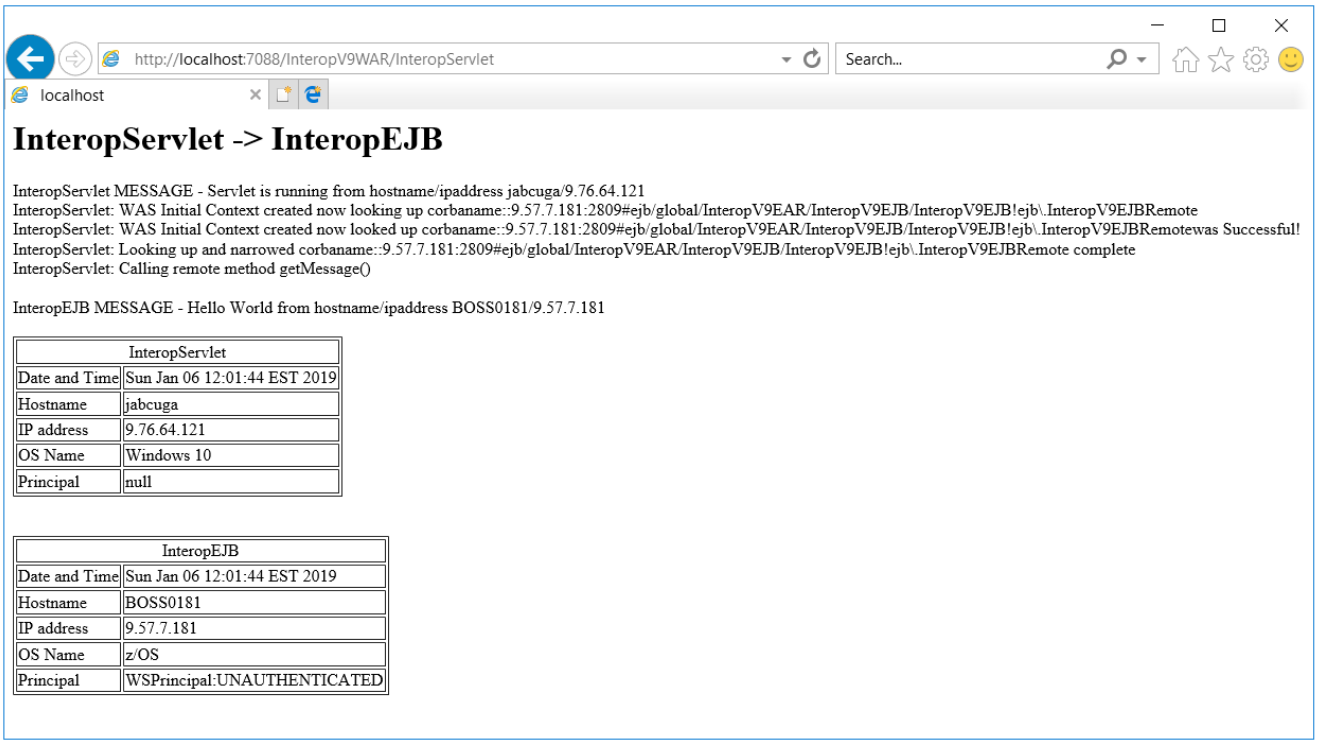
```
<server description="LiberyZOS">

<featureManager>
        <feature>jsp-2.3</feature>
        <feature>servlet-4.0</feature>
        <feature>ejb-3.2</feature>
        <feature>appSecurity-2.0</feature>
        <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="boss0181.pok.ibm.com" id="defaultIiopEndpoint" iiopPort="2809">
        <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
        clientAuthentication="false" clientAuthenticationSupported="false"  serverKeyAlias="ZosPersonal"/>

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/ZosKeystore.jks"
        password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/ZosTruststore.jks"
        password="Liberty" type="JKS" />

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never"/>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
        <security-role name="AuthorizedUser">
                <user access-id="keith" name="keith"/>
        </security-role>
  </application-bnd>
</enterpriseApplication>

<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>
</server>
```

Windows to z/OS – working scenario

When the servlet on Liberty on Windows calls the Liberty on z/OS server, the non-SSL iiopPort of 2809 is used to call the EJB. Although its not seen in the application, the Liberty Server will perform an SSL call to the Liberty z/OS server over the iiopPorts port.

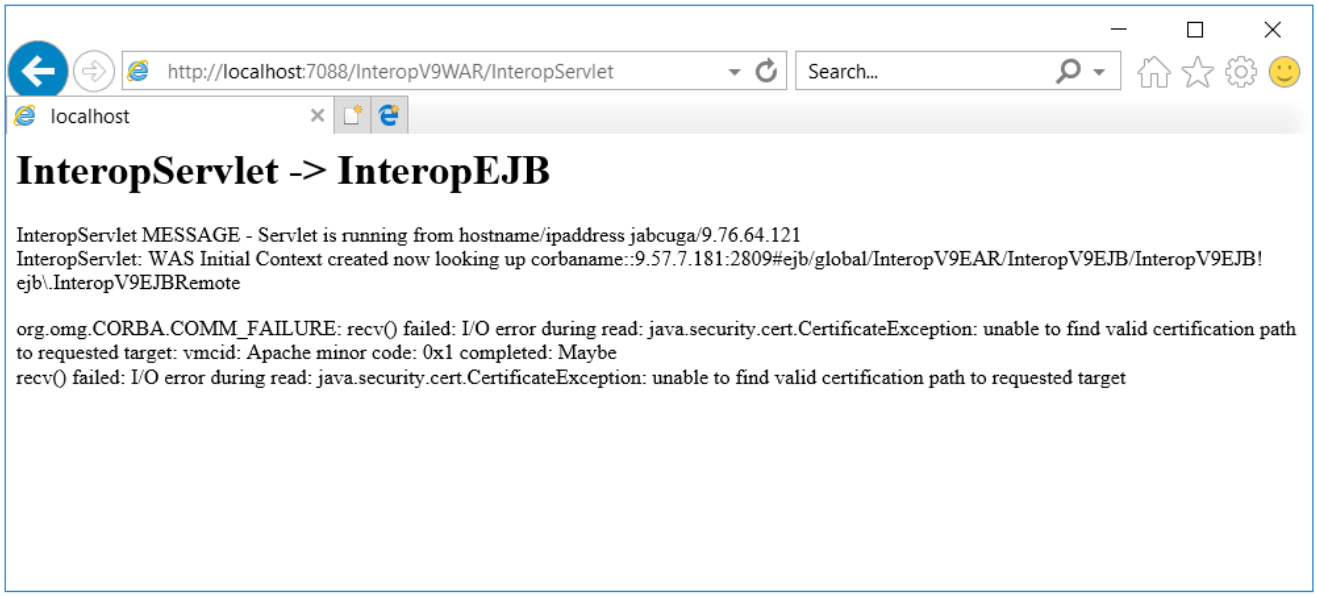


The successful output looks the same as the scenario without SSL



Windows to z/OS – Failing scenario

If the client Liberty Windows server does not have the correct signing certificate in its truststore that signed the personal certicate received from the Liberty z/OS server's keystore an SSL error will surface in both the application as well as in the messages.log of the client Liberty Windows Server.



The messages.log will show

[ERROR] CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN CN=boss0181.pok.ibm.com was sent from the target host. **The signer might need to be added to local trust store C:/Program1/Liberty18004/wlp/usr/servers/defaultServer/resources/security/WinTruststore.jks**, located in SSL configuration alias DefaultSSLSettings. The extended error message from the SSL handshake exception is: PKIX path building failed: java.security.cert.CertPathBuilderException: unable to find valid certification path to requested target org.omg.CORBA.COMM_FAILURE: recv() failed: I/O error during read: java.security.cert.CertificateException: unable to find valid certification path to requested target: vmcid: Apache minor code: 0x1 completed: Maybe/nrecv() failed: I/O error during read: java.security.cert.CertificateException: unable to find valid certification path to requested target
[err] org.omg.CORBA.COMM_FAILURE: recv() failed: I/O error during read: java.security.cert.CertificateException: unable to find valid certification path to requested target: vmcid: Apache minor code: 0x1 completed: Maybe
[err] at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
[err] at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:88)
[err] at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:57)
[err] at java.lang.reflect.Constructor.newInstance(Constructor.java:437)
[err] at org.apache.yoko.orb.OB.Util.copySystemException(Util.java:81)
[err] at org.apache.yoko.orb.OB.GIOPConnectionThreaded.access$200(GIOPConnectionThreaded.java:42)
[err] at org.apache.yoko.orb.OB.GIOPConnectionThreaded$Receiver.run(GIOPConnectionThreaded.java:68)
[err] at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:522)
[err] at java.util.concurrent.FutureTask.run(FutureTask.java:277)
[err] at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1153)
[err] at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
[err] at java.lang.Thread.run(Thread.java:785)

CSIv2 Authentication Layer – No Trust

In the prior section “CSIv2 Transport Layer - Securing the transport with SSL”
The Liberty server on Windows and Liberty Server on z/OS both had trust disabled.

```
<authenticationLayer establishTrustInClient="Never"/>
```

What this means is that there is no trust established between the Liberty Windows Server and the Liberty z/OS server. Any client Liberty server could make a call into the Liberty z/OS server without having to authenticate to the Liberty z/OS server.

```
<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never"/>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>
```

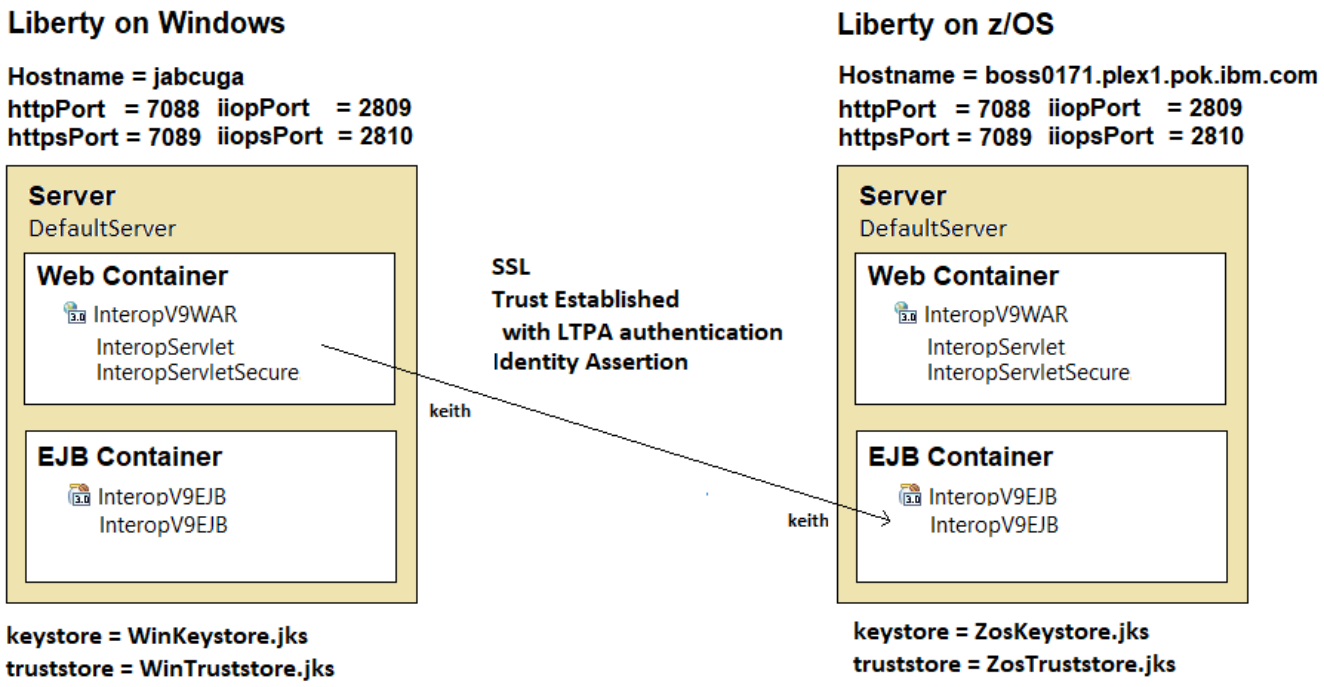
In the next two sections, the serverPolicy.csiv2 tag will be set to:

```
<authenticationLayer establishTrustInClient="Required"/>
```

so that the client Liberty Windows server will be required to authenticate to the remote Liberty z/OS in order to establish the trust relationship between the two servers.

CSIv2 Attribute Layer – Identity Assertion using LTPA Authentication for Trust

In this scenario, the Liberty for z/OS server will establish trust using LTPA authentication, but will also be able to assert the unauthenticated identity or a user identity (ie. keith) from the client Liberty Windows Server to Liberty for z/OS.



The ltpa.keys file is created the first time the server is started and is located in Liberty Windows and Liberty z/OS servers directory

```
${server.config.dir}/resources/security/ltpa.keys
```

To ensure both servers have the same LTPA keys. Copy this file from one of the Liberty servers (ie. Liberty on z/OS) to the other Liberty server (ie. Liberty on Windows).

The Liberty Server on z/OS serverPolicy.csiv2 tag will have:

```
<authenticationLayer mechanisms="LTPA" establishTrustInClient="Required">
```

The Liberty Server on Windows clientPolicy.csiv2 tag will have:

```
<authenticationLayer mechanisms="LTPA" establishTrustInClient="Supported">
```

server.xml – Liberty on Windows

```
<server description="LibertyWindows">

<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="jabcuga" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
            clientAuthentication="false" clientAuthenticationSupported="false" />

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/WinKeystore.jks"
            password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/WinTruststore.jks"
            password="Liberty" type="JKS" />

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true" trustedIdentities="*">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="LTPA" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="LTPA" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>
```

```
<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>

</server>
```

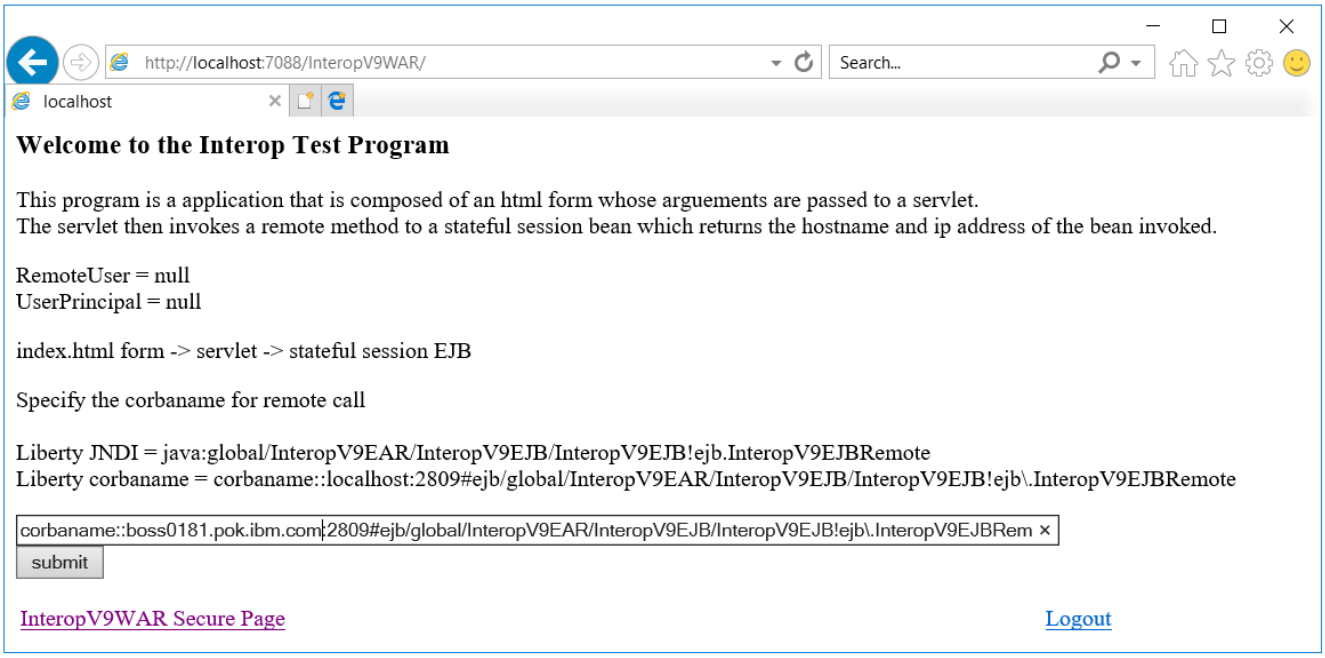
server.xml – Liberty on z/OS

```
<server description="LiberyZOS">

<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="boss0181.pok.ibm.com" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
            clientAuthentication="false" clientAuthenticationSupported="false"  serverKeyAlias="ZosPersonal"/>

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/ZosKeystore.jks"
            password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/ZosTruststore.jks"
            password="Liberty" type="JKS" />

orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true" trustedIdentities="*">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="LTPA" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="LTPA" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>
```

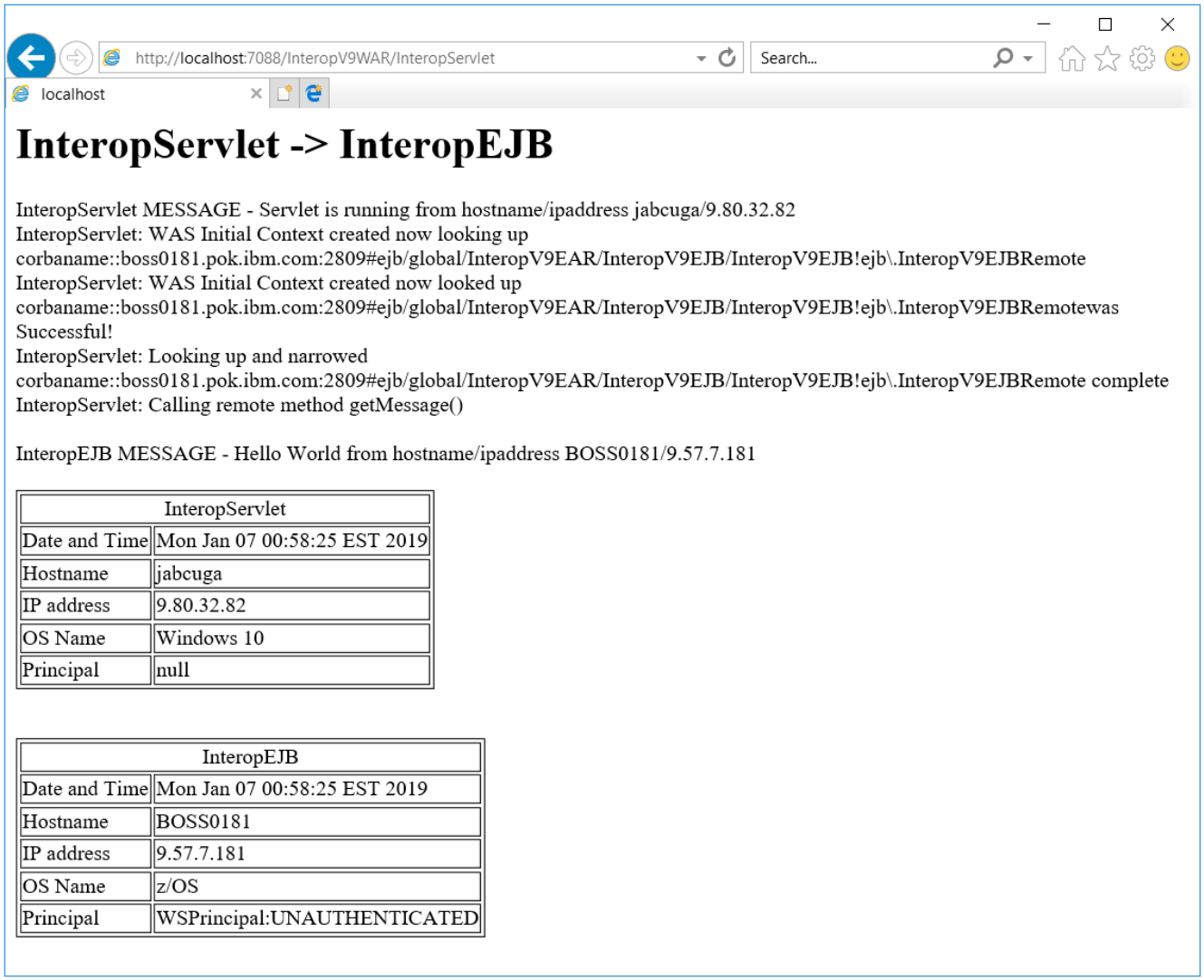
```
<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>
</server>
```

Windows to z/OS – working scenario 1 (unauthenticated)

In this scenario the client is running unauthenticated.



After successful call, the unauthenticated identity is running on the thread in the remote Liberty for z/OS server.



InteropServlet -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

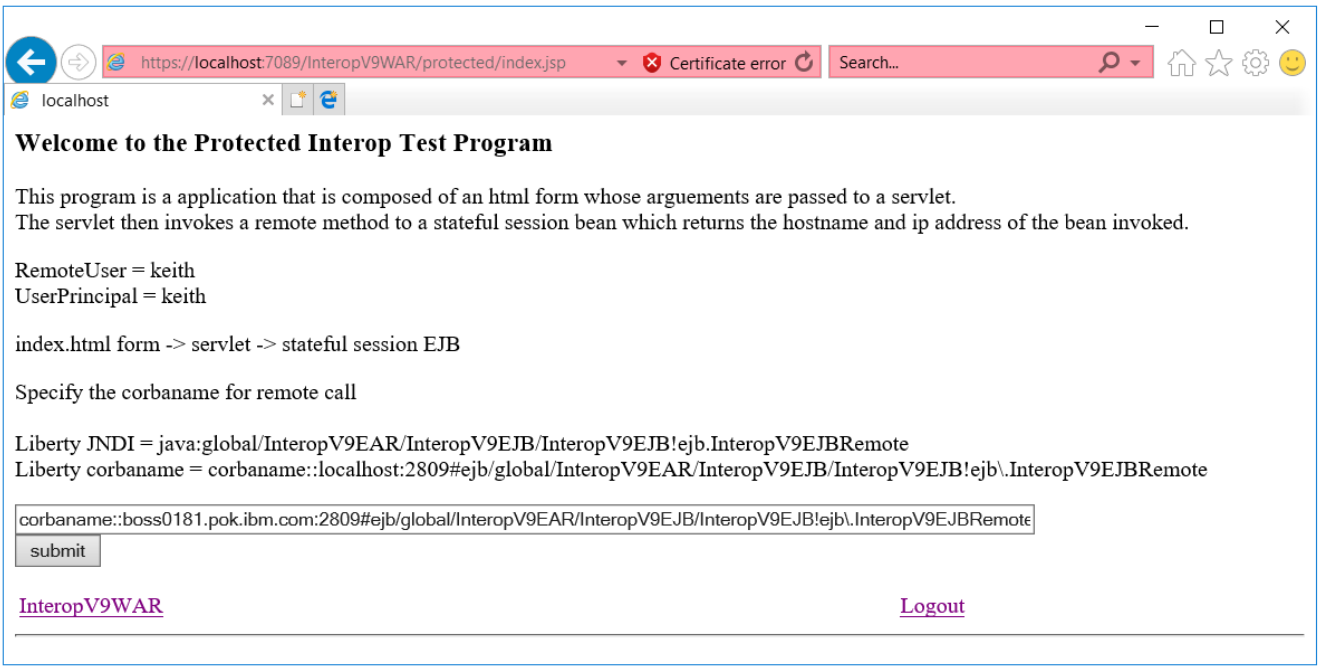
InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

InteropServlet	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	null

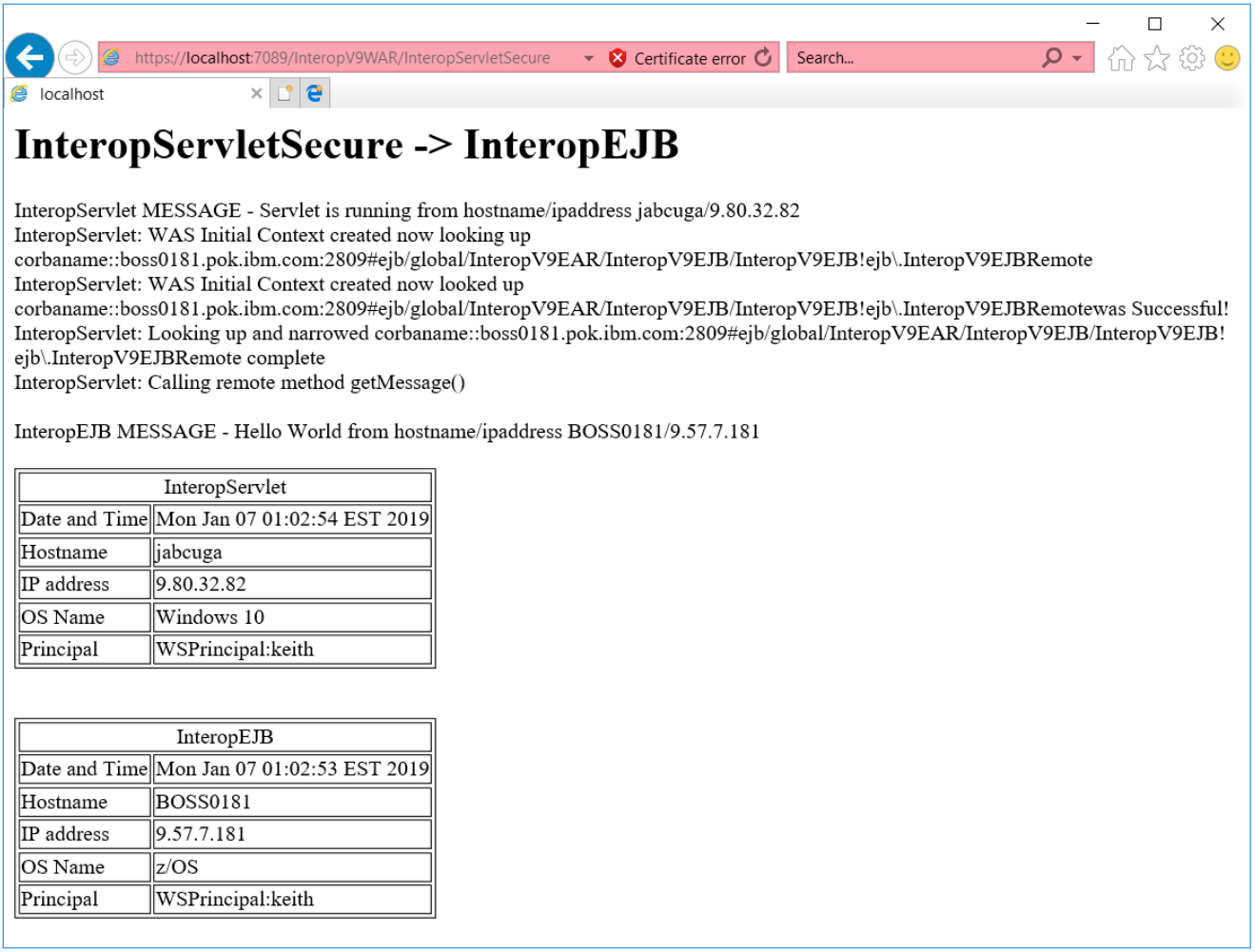
InteropEJB	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:UNAUTHENTICATED

Windows to z/OS – working scenario 2 (authenticated)

In this scenario the client is authenticated. User “keith” has logged in on the form based web application.



After successful login, the identity “keith” is running on the thread in the remote Liberty for z/OS server.



InteropServletSecure -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!
ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

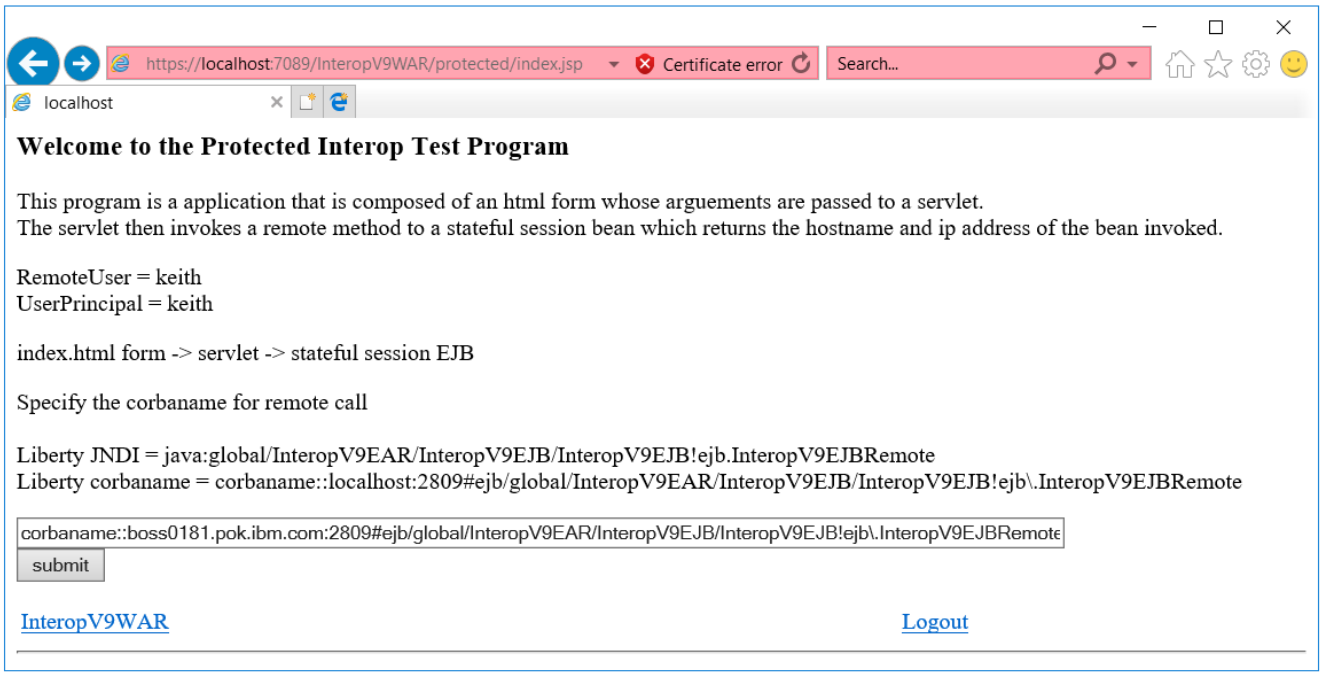
InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

InteropServlet	
Date and Time	Mon Jan 07 01:02:54 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	WSPrincipal:keith

InteropEJB	
Date and Time	Mon Jan 07 01:02:53 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:keith

Windows to z/OS – Failing scenario

If the user logs in on the client Liberty Windows server, but the LTPA keys are not exchanged between servers such that they do not have a common LTPA key.



The client will receive a NO_PERMISSION error.

Liberty for z/OS messages.log will show:

[1/7/19 0:15:22:134 EST] 0000050f com.ibm.ws.transport.iiop.security.ServerSecurityInterceptor E SASException com.ibm.ws.transport.iiop.security.SASInvalidEvidenceException: org.omg.CORBA.NO_PERMISSION: CWWKS4001I: The security token cannot be validated. This can be for the following reasons
1. The security token was generated on another server using different keys.
2. The token configuration or the security keys of the token service which created the token has been changed.
3. The token service which created the token is no longer available.: vmcid: 0x49424000 minor code: 0x300 completed: Nocom.ibm.ws.security.csiv2.server.config.tss.**ServerLTPAMechConfig.check**(ServerLTPAMechConfig.java:110)com.ibm.ws.transport.iiop.security.config.tss.TSSCompoundSecMechConfig.check(TSSCompoundSecMechConfig.java:145)com.ibm.ws.transport.iiop.security.config.tss.TSSCompoundSecMechListConfig.check(TSSCompoundSecMechListConfig.java:109)
com.ibm.ws.transport.iiop.security.config.tss.TSSConfig.check(TSSConfig.java:56)
com.ibm.ws.transport.iiop.security.ServerSecurityInterceptor.receive_request(ServerSecurityInterceptor.java:138)
org.apache.yoko.orb.PortableInterceptor.ServerRequestInfo_impl._OB_request(ServerRequestInfo_impl.java:419)
org.apache.yoko.orb.OB.PIManager.serverReceiveRequest(PIManager.java:439)
org.apache.yoko.orb.OB.PIUpcall.postUnmarshal(PIUpcall.java:86)
org.apache.yoko.orb.OBPortableServer.ServantDispatcher.dispatchBase(ServantDispatcher.java:111)
org.apache.yoko.orb.OBPortableServer.ServantDispatcher.dispatch(ServantDispatcher.java:148)
org.apache.yoko.orb.OBPortableServer.POA_impl._OB_dispatch(POA_impl.java:1644)
org.apache.yoko.orb.OB.DispatchRequest_impl.invoke(DispatchRequest_impl.java:56)
com.ibm.ws.transport.iiop.yoko.ExecutorDispatchStrategy$1.run(ExecutorDispatchStrategy.java:42)
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1160)
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635)
java.lang.Thread.run(Thread.java:812)

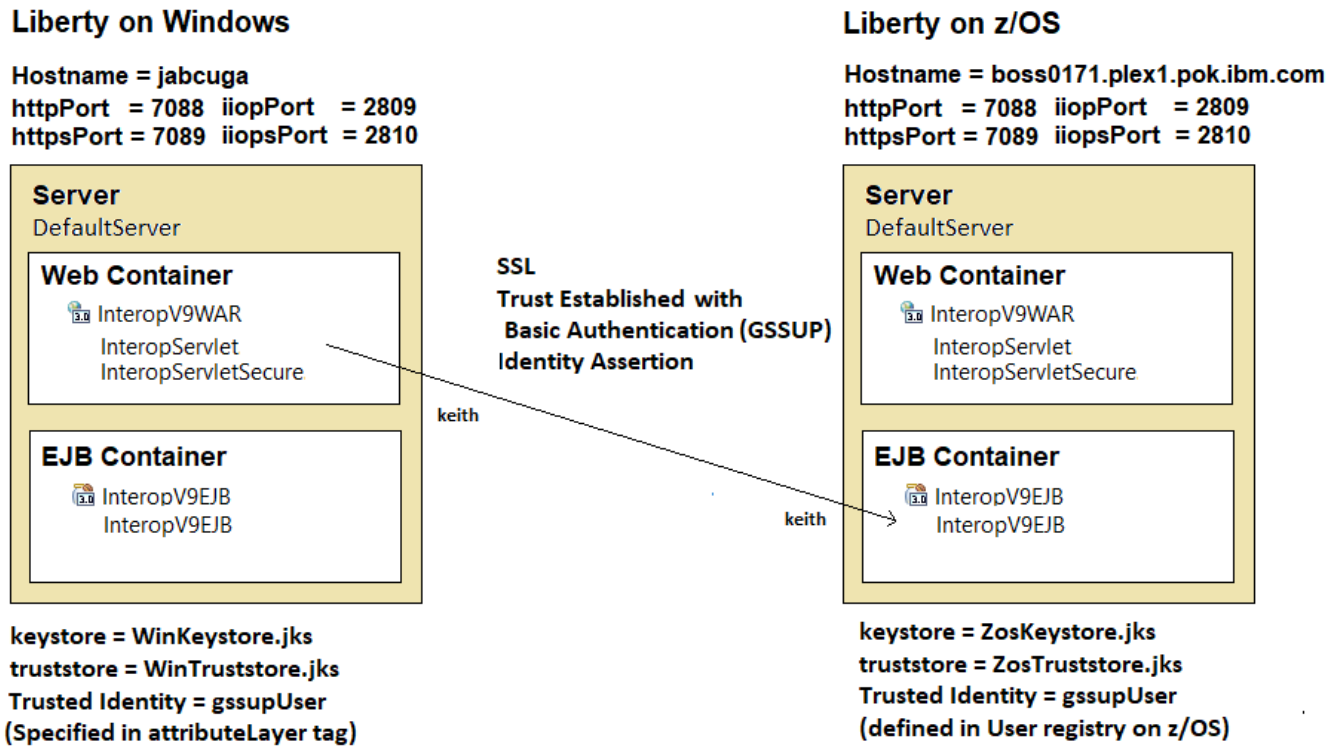
Caused by: org.omg.CORBA.NO_PERMISSION: CWWKS4001I: The security token cannot be validated.
This can be for the following reasons
1. The security token was generated on another server using different keys.
2. The token configuration or the security keys of the token service which created the token has been changed.
3. The token service which created the token is no longer available.: vmcid: 0x49424000 minor code: 0x300 completed: No
com.ibm.ws.transport.iiop.security.SASInvalidEvidenceException.<init>(SASInvalidEvidenceException.java:32)
... 16 more

The Liberty Server on Windows messages.log will show:

SERVLET MESSAGE - is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
org.omg.**CORBA.NO_PERMISSION**: : vmcid: 0x49424000 minor code: 0x300 completed: No/n
[err] org.omg.CORBA.NO_PERMISSION: : vmcid: 0x49424000 minor code: 0x300 completed: No
[err] at org.apache.yoko.orb.OB.Util.unmarshalSystemException(Util.java:124)
[err] at [internal classes]
[err] at org.apache.yoko.orb.OB.GIOPConnectionThreaded.access$200(GIOPConnectionThreaded.java:42)
[err] at org.apache.yoko.orb.OB.GIOPConnectionThreaded$Receiver.run(GIOPConnectionThreaded.java:68)
[err] at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:522)
[err] at java.util.concurrent.FutureTask.run(FutureTask.java:277)
[err] at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1153)
[err] at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:628)
[err] at java.lang.Thread.run(Thread.java:785)

CSIv2 Attribute Layer – Identity Assertion using Basic Authentication (GSSUP) for Trust

In this scenario, the Liberty for z/OS server will establish trust using Basic Authentication, and also assert the unauthenticated identity or a user identity (ie. keith) from the client Liberty Windows Server to Liberty for z/OS.



server.xml – Liberty on Windows

```
<server description="LibertyWindows">

<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="jabcuga" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
            clientAuthentication="false" clientAuthenticationSupported="false" />

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/WinKeystore.jks"
            password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/WinTruststore.jks"
            password="Liberty" type="JKS" />

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true" trustedIdentities="*">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
        <attributeLayer identityAssertionEnabled="true"
          trustedIdentity="gssupUser" trustedPassword="gssupPassword">
          <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
          <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>
```

```
<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>

</server>
```

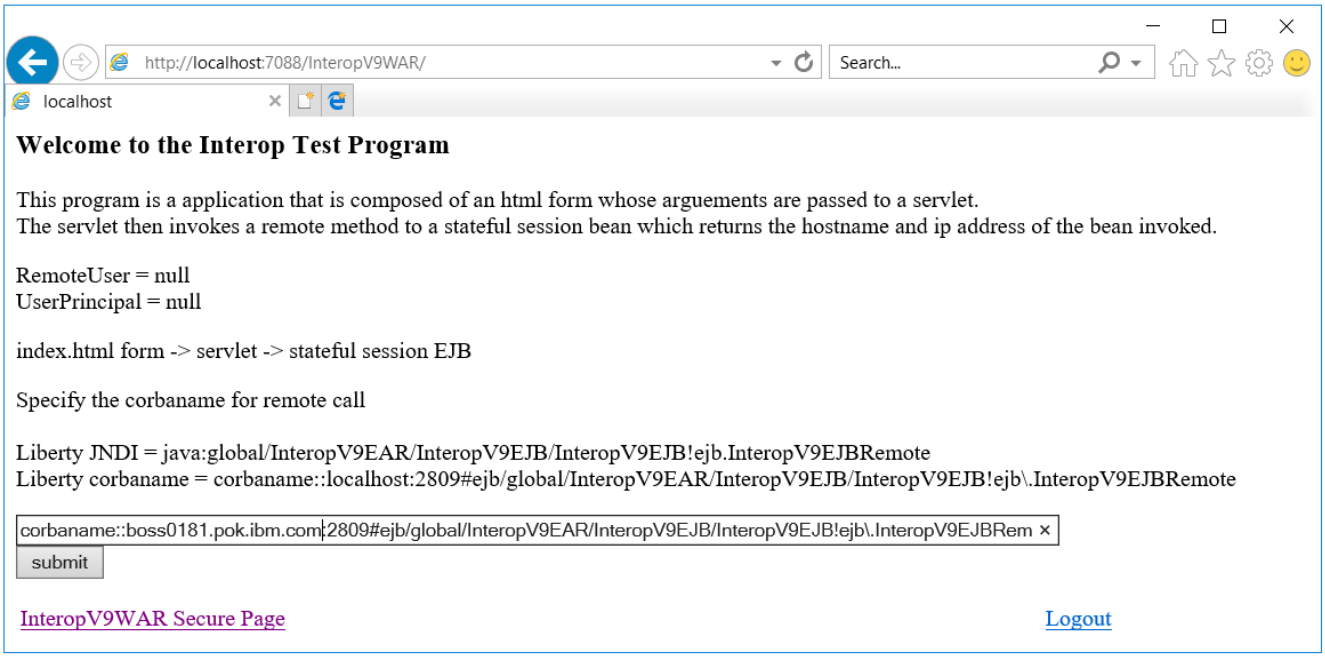
server.xml – Liberty on z/OS

```
<server description="LiberyZOS">

<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="boss0181.pok.ibm.com" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"
            clientAuthentication="false" clientAuthenticationSupported="false"  serverKeyAlias="ZosPersonal"/>

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/ZosKeystore.jks"
            password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/ZosTruststore.jks"
            password="Liberty" type="JKS" />

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true" trustedIdentities="*">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true" >
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer mechanisms="GSSUP" establishTrustInClient="Required">
         </authenticationLayer>
         <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
      </layers>
   </clientPolicy.csiv2>
</orb>

<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>
```

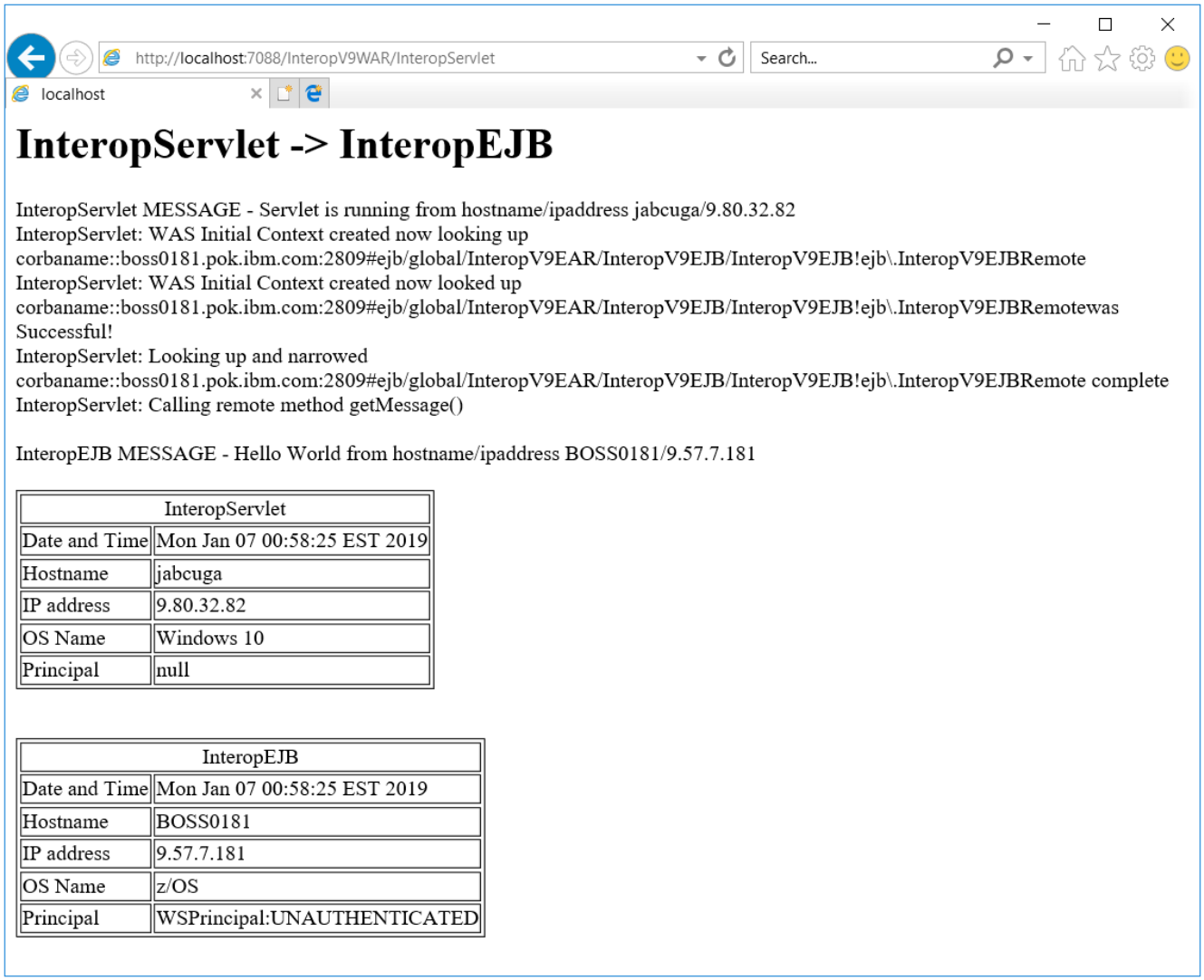
```
<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
    <user name="gssupUser" password="gssupPassword"/>
</basicRegistry>
</server>
```

Windows to z/OS – working scenario 1 (unauthenticated)

In this scenario the client is running unauthenticated.



After successful call, the unauthenticated identity is running on the thread in the remote Liberty for z/OS server.



InteropServlet -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

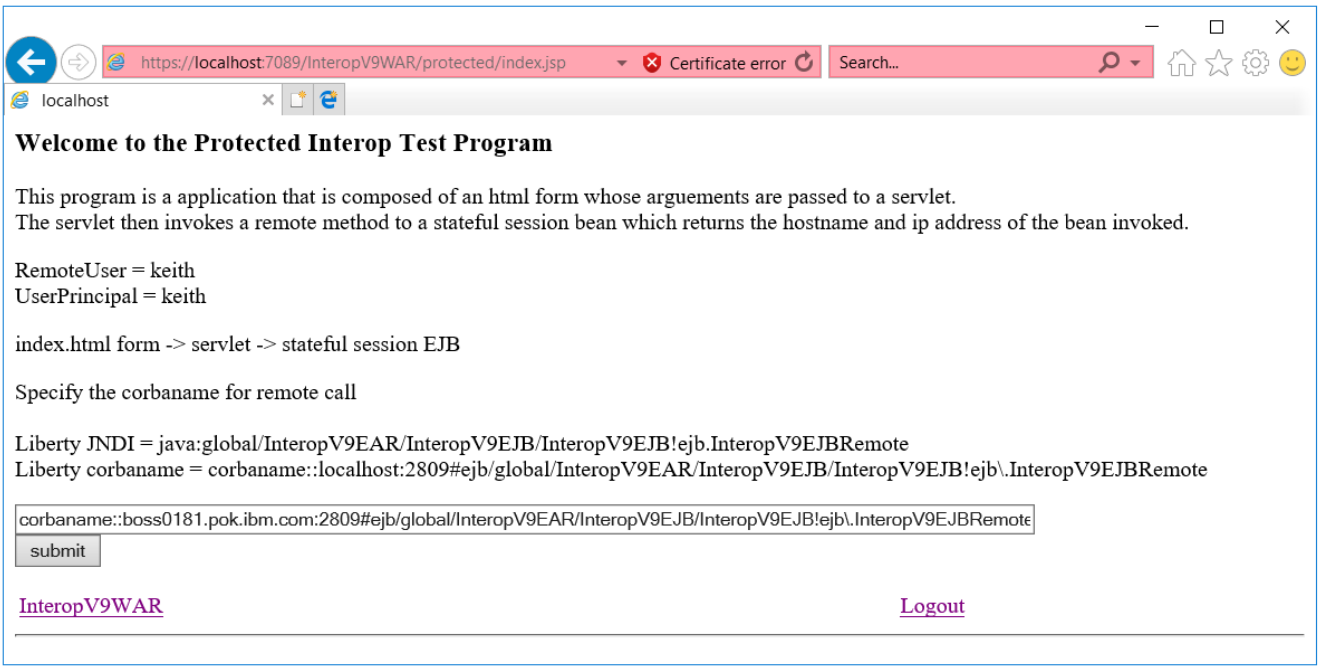
InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

InteropServlet	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	null

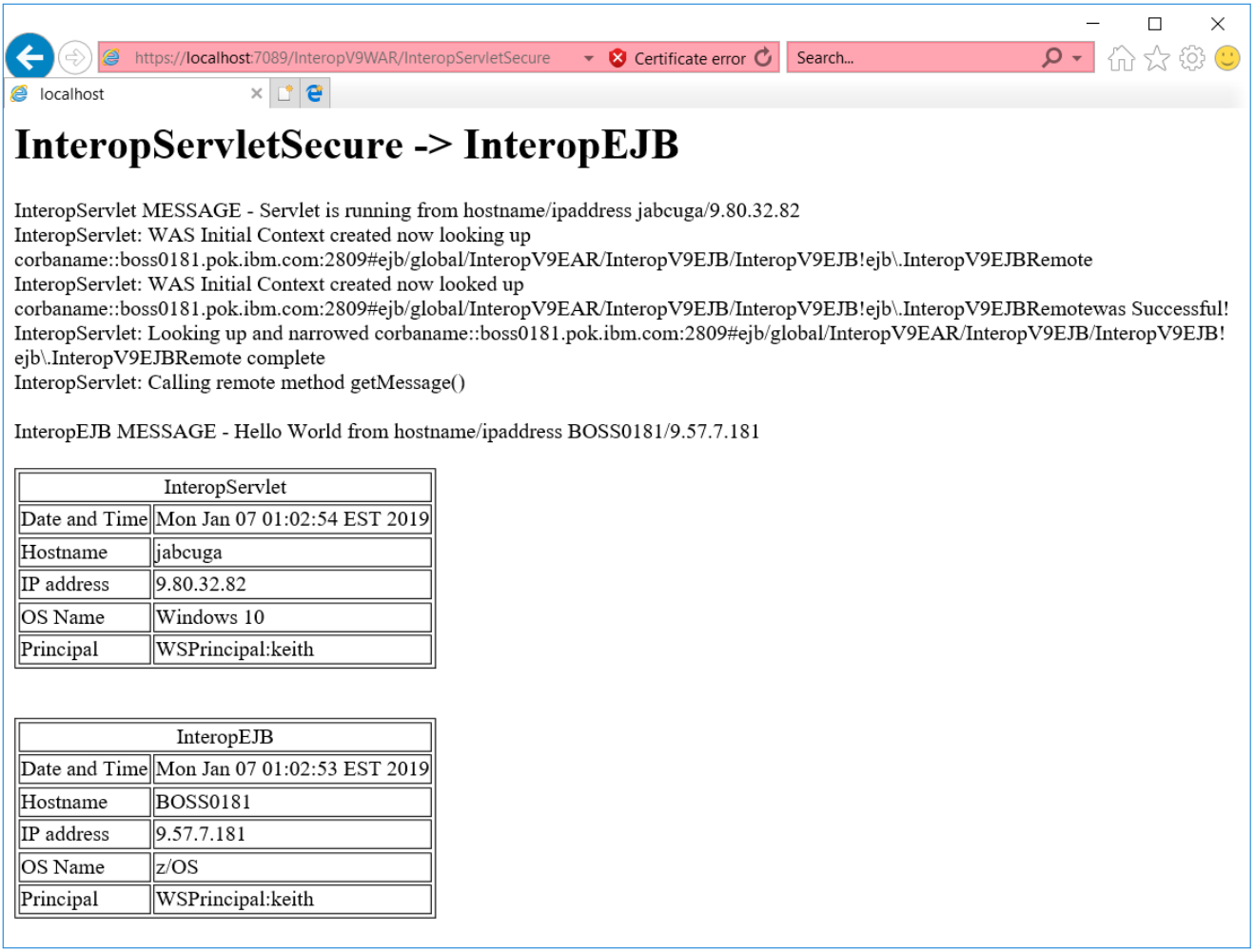
InteropEJB	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:UNAUTHENTICATED

Windows to z/OS – working scenario 2 (authenticated)

In this scenario the client is authenticated. User “keith” has logged in on the form based web application.



After successful login, the identity “keith” is running on the thread in the remote Liberty for z/OS server.



InteropServletSecure -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!
ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

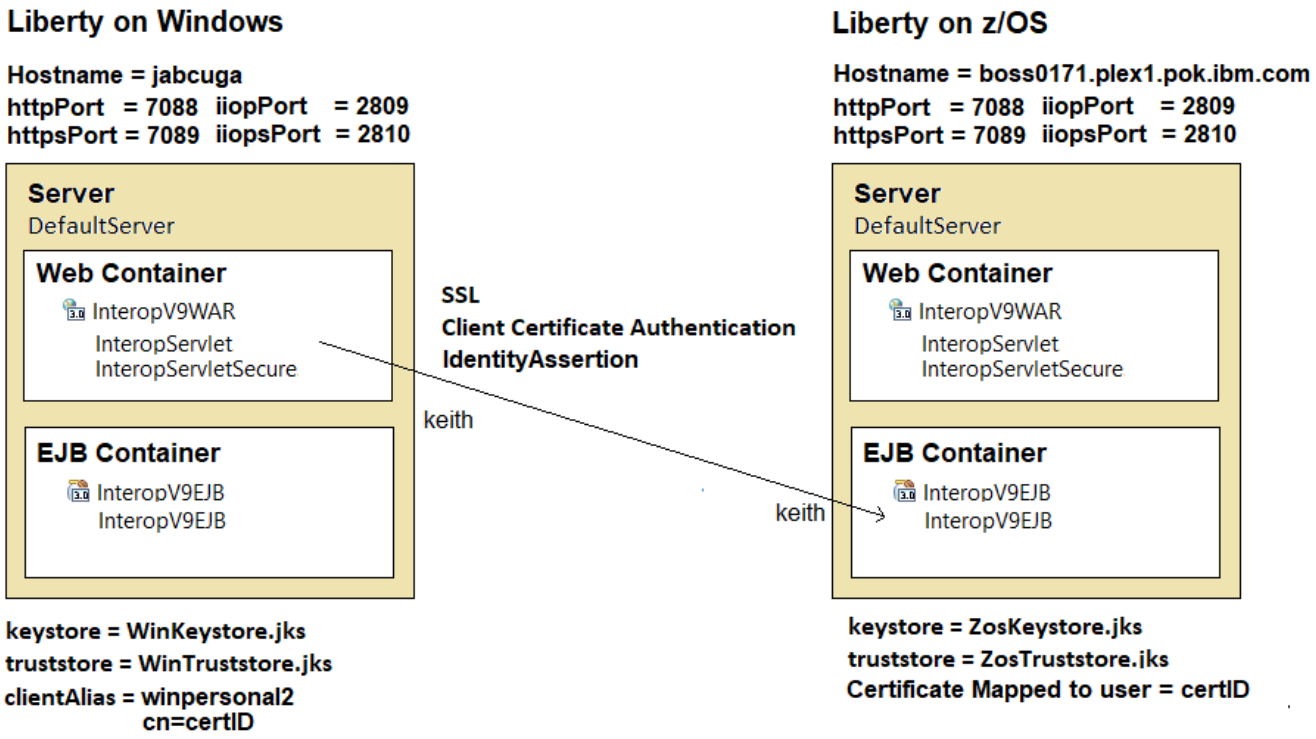
InteropServlet	
Date and Time	Mon Jan 07 01:02:54 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	WSPrincipal:keith

InteropEJB	
Date and Time	Mon Jan 07 01:02:53 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:keith

CSIv2 Attribute Layer – Identity Assertion using Client Certificate Authentication

Once the InteropV9 application has been tested cross platform successfully with security enabled using SSL to secure the transport, a new personal certificate can be created to perform client certificate authentication. The Liberty z/OS server.xml can be enabled to require clientAuthentication, and the Liberty Windows server.xml can be set to support clientAuthentication.

The serverPolicy.csiv2 and clientPolicy.csiv2 tags are set in the server.xml to disable establishing trust between the two Liberty servers since there is no LTPA token or GSSUP token. The Liberty Windows Server will assert the unauthenticated identity or a user identity (ie. keith) from the client Liberty Windows Server to Liberty for z/OS.



The new personal certificate that is created is installed in the Liberty on Windows keystore to be used for client certificate authentication to the Liberty for z/OS Server. The personal certificate winpersonal2 has a distinguished name of cn=certID. The signer certificate that signed winpersonal2 must exist in the z/OS Liberty server truststore, and the user “certID” must exist in the z/OS user registry.

This section assumes the openssl and keytool commands for creating the kestore and truststore and their corresponding certificates were issued in
“Using a Personal Certificate Signed by a Certificate Authority”
from the section
“CSIv2 Transport Layer - Securing the transport with SSL”

OpenSSL commands to sign a second Personal certificate

In this scenario, the openssl commands below are used to create a second personal certificate signed by the same signer certificate “WinRootSigner” used in the section “Using a Personal Certificate Signed by a Certificate Authority”
Note that the subject CN contains a user that is a valid user in the Liberty for z/OS basicRegistry.

Generate Server Certificate signed by the CA Certificate

Generate private Key
openssl genrsa -out WinPersonal2.key 2048

Generate Personal Certificate Signing Request (CSR)
openssl req -new -sha256 -key WinPersonal2.key -out WinPersonal2.csr -subj "/CN=certID"

Generate Personal Certificate
openssl x509 -req -in WinPersonal2.csr -CA WinRootSigner.pem -CAkey WinRootSigner.key -CAcreateserial -out WinPersonal2.crt -days 3650 -sha256

View the certificates
openssl x509 -text -noout -in WinPersonal2.crt

Convert Personal PEM certificate and a private key to password protected PKCS#12 (.p12) format
openssl pkcs12 -export -out WinPersonal2.p12 -inkey WinPersonal2.key -in WinPersonal2.crt -certfile WinRootSigner.pem -passout pass:Liberty -alias “WinPersonal2”

Keytool Commands to import the Signer and Personal Certificate

The keytool command is used to import the second personal certificate “WinPersonal2” into with the openssl commands into the keystore WinKeyStore.jks.

Import the personal certificate and private key from a password protected pkcs12 file into keystore
keytool -importkeystore -deststorepass Liberty -destkeystore WinKeystore.jks -srcstorepass Liberty -srckeystore WinPersonal2.p12 -srcstoretype PKCS12 -alias WinPersonal2

Check the contents of the Windows keystore and truststore
keytool -list -v -keystore WinKeystore.jks -storepass Liberty

Windows keystore

The commands below show the contents of the Windows truststore file.

```
$ keytool -list -v -keystore WinKeystore.jks -storepass Liberty

Keystore type: jks
Keystore provider: IBMJCE
Your keystore contains 2 entries

Alias name: winpersonal2
Creation date: Feb 4, 2019
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=certID
Issuer: CN=WinRootSigner
Serial number: b78e5287624fbf94
Valid from: 1/30/19 4:33 PM until: 1/27/29 4:33 PM
Certificate fingerprints:
        MD5:  D1:A1:F0:62:0D:4C:23:F7:4C:7A:A7:8C:65:11:2B:E5
        SHA1: E3:D0:C1:6F:6C:90:66:99:39:3D:39:93:F5:12:1A:7B:71:C3:E3:BE
        SHA256: 07:65:05:DD:B3:BA:78:2A:F5:73:3B:AC:69:72:18:74:5C:E9:82:BF:88:
72:FE:B2:D3:EF:88:13:7B:D9:01:23
        Signature algorithm name: SHA256withRSA
        Version: 1
Certificate[2]:
Owner: CN=WinRootSigner
Issuer: CN=WinRootSigner
Serial number: cb55c54e1c7be635
Valid from: 1/6/19 1:40 PM until: 1/3/29 1:40 PM
Certificate fingerprints:
        MD5:  57:76:D3:68:D7:22:3C:54:B9:46:6B:5B:EC:95:15:D8
        SHA1: C3:63:A0:5E:CD:7D:85:5A:17:F8:3F:E7:79:62:EB:DC:3D:BF:AB:1D
        SHA256: 41:B5:07:FC:8E:1A:6F:BB:40:87:79:CE:06:64:B4:F6:FE:01:BF:01:54:
A2:11:B5:C8:C7:52:06:51:53:D9:8E
        Signature algorithm name: SHA256withRSA
        Version: 1
*******************************************
Alias name: winpersonal
Creation date: Jan 6, 2019
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=jabcuga
Issuer: CN=WinRootSigner
Serial number: b78e5287624fbf93
Valid from: 1/6/19 1:41 PM until: 1/3/29 1:41 PM
Certificate fingerprints:
        MD5:  57:3F:58:13:45:58:B8:2E:48:DE:C3:88:CE:75:15:F0
        SHA1: 04:EF:4B:97:68:F3:13:BE:95:BC:32:BB:97:EF:F3:2B:03:EF:97:C8
        SHA256: 8E:8C:B3:E1:4E:42:61:CB:7D:82:C4:EF:82:C5:49:5D:9A:53:F7:04:4D:
15:53:E9:99:E4:7F:C3:CE:48:2D:11
        Signature algorithm name: SHA256withRSA
        Version: 1
Certificate[2]:
```

Owner: CN=WinRootSigner
Issuer: CN=WinRootSigner
Serial number: cb55c54e1c7be635
Valid from: 1/6/19 1:40 PM until: 1/3/29 1:40 PM
Certificate fingerprints:
	MD5: 57:76:D3:68:D7:22:3C:54:B9:46:6B:5B:EC:95:15:D8
	SHA1: C3:63:A0:5E:CD:7D:85:5A:17:F8:3F:E7:79:62:EB:DC:3D:BF:AB:1D
	SHA256: 41:B5:07:FC:8E:1A:6F:BB:40:87:79:CE:06:64:B4:F6:FE:01:BF:01:54:
A2:11:B5:C8:C7:52:06:51:53:D9:8E
	Signature algorithm name: SHA256withRSA
	Version: 1

****

server.xml – Liberty on Windows

```
<server description="LibertyWindows">
<featureManager>
            <feature>jsp-2.3</feature>
            <feature>servlet-4.0</feature>
            <feature>ejb-3.2</feature>
            <feature>appSecurity-2.0</feature>
            <feature>ssl-1.0</feature>
   </featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint">
</httpEndpoint>

<iiopEndpoint host="jabcuga" id="defaultIiopEndpoint" iiopPort="2809">
            <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl clientAuthentication="false" clientAuthenticationSupported="true" clientKeyAlias="winpersonal2"
id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore" trustStoreRef="CellDefaultTrustStore"/>

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/WinKeystore.jks" password="Liberty"
type="JKS"/>
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/WinTruststore.jks" password="Liberty"
type="JKS"/>

<orb id="defaultOrb">
   <serverPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="false">
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never">
         </authenticationLayer>
         <transportLayer sslEnabled="true"/>
      </layers>
   </serverPolicy.csiv2>
   <clientPolicy.csiv2>
      <layers>
         <attributeLayer identityAssertionEnabled="true">
           <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
           <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
         </attributeLayer>
         <authenticationLayer establishTrustInClient="Never"/>
         <transportLayer sslEnabled="true"/>
      </layers>
   </clientPolicy.csiv2>
</orb>


<enterpriseApplication id="InteropV9EAR" location="InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
            <security-role name="AuthorizedUser">
                        <user access-id="keith" name="keith"/>
            </security-role>
  </application-bnd>
</enterpriseApplication>

<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
</basicRegistry>
</server>
```

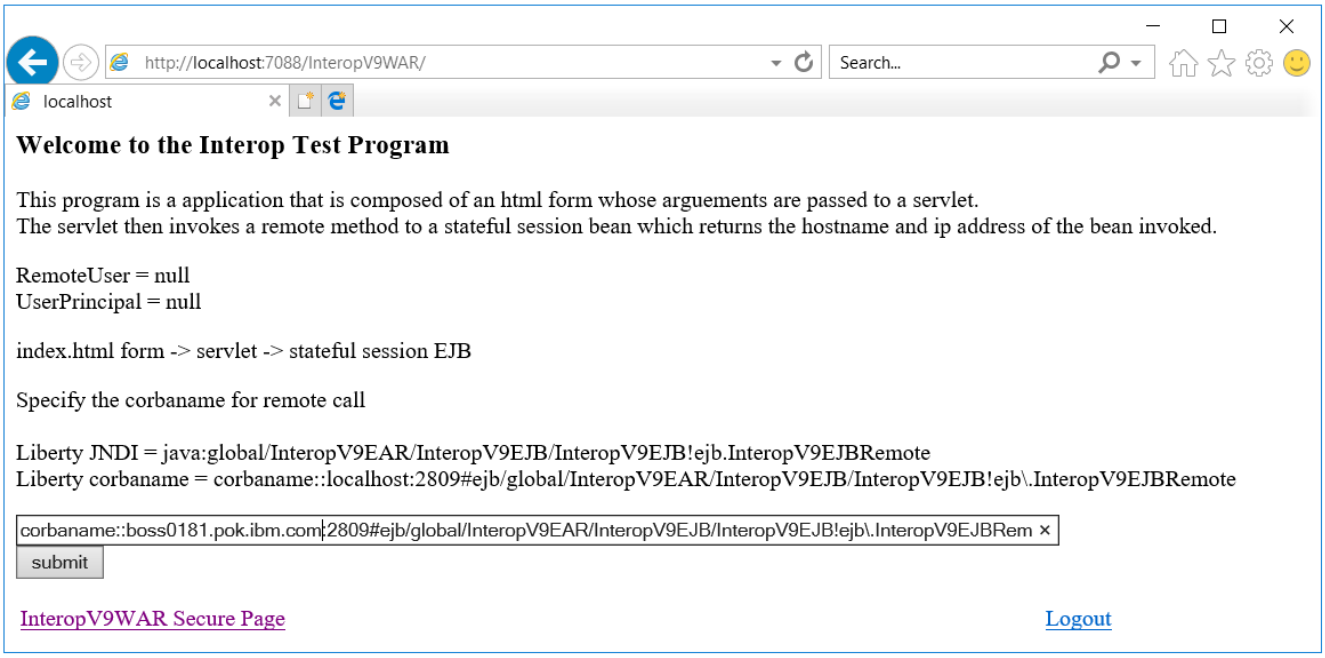
server.xml – Liberty on z/OS

```
<server description="LiberyZOS">

<featureManager>
        <feature>jsp-2.3</feature>
        <feature>servlet-4.0</feature>
        <feature>ejb-3.2</feature>
        <feature>appSecurity-2.0</feature>
        <feature>ssl-1.0</feature>
</featureManager>

<httpEndpoint httpPort="7088" httpsPort="7089" id="defaultHttpEndpoint" host="*">
</httpEndpoint>

<iiopEndpoint host="boss0181.pok.ibm.com" id="defaultIiopEndpoint" iiopPort="2809">
        <iiopsOptions iiopsPort="2810" sslRef="DefaultSSLSettings"/>
</iiopEndpoint>

<sslDefault sslRef="DefaultSSLSettings"/>

<ssl id="DefaultSSLSettings" keyStoreRef="CellDefaultKeyStore"
trustStoreRef="CellDefaultTrustStore" clientAuthentication="true"  serverKeyAlias="ZosPersonal"/> />

<keyStore id="CellDefaultKeyStore" location="${server.config.dir}/resources/security/ZosKeystore.jks"
        password="Liberty"  type="JKS" />
<keyStore id="CellDefaultTrustStore" location="${server.config.dir}/resources/security/ZosTruststore.jks"
        password="Liberty" type="JKS" />

<orb id="defaultOrb">
    <serverPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="true">
              <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
              <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
            </attributeLayer>
            <authenticationLayer establishTrustInClient="Never">
            </authenticationLayer>
            <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
        </layers>
    </serverPolicy.csiv2>
    <clientPolicy.csiv2>
        <layers>
            <attributeLayer identityAssertionEnabled="false">
                <identityAssertionTypes>ITTAnonymous</identityAssertionTypes>
                <identityAssertionTypes>ITTPrincipalName</identityAssertionTypes>
            </attributeLayer>
            <authenticationLayer establishTrustInClient="Never">
             </authenticationLayer>
            <transportLayer sslEnabled="true" sslRef="DefaultSSLSettings"/>
        </layers>
    </clientPolicy.csiv2>
        </layers>
    </clientPolicy.csiv2>
</orb>


<enterpriseApplication id="InteropV9EAR" location="${server.config.dir}/InteropV9EAR.ear" name="InteropV9EAR">
  <application-bnd>
        <security-role name="AuthorizedUser">
              <user access-id="keith" name="keith"/>
        </security-role>
  </application-bnd>
</enterpriseApplication>
```

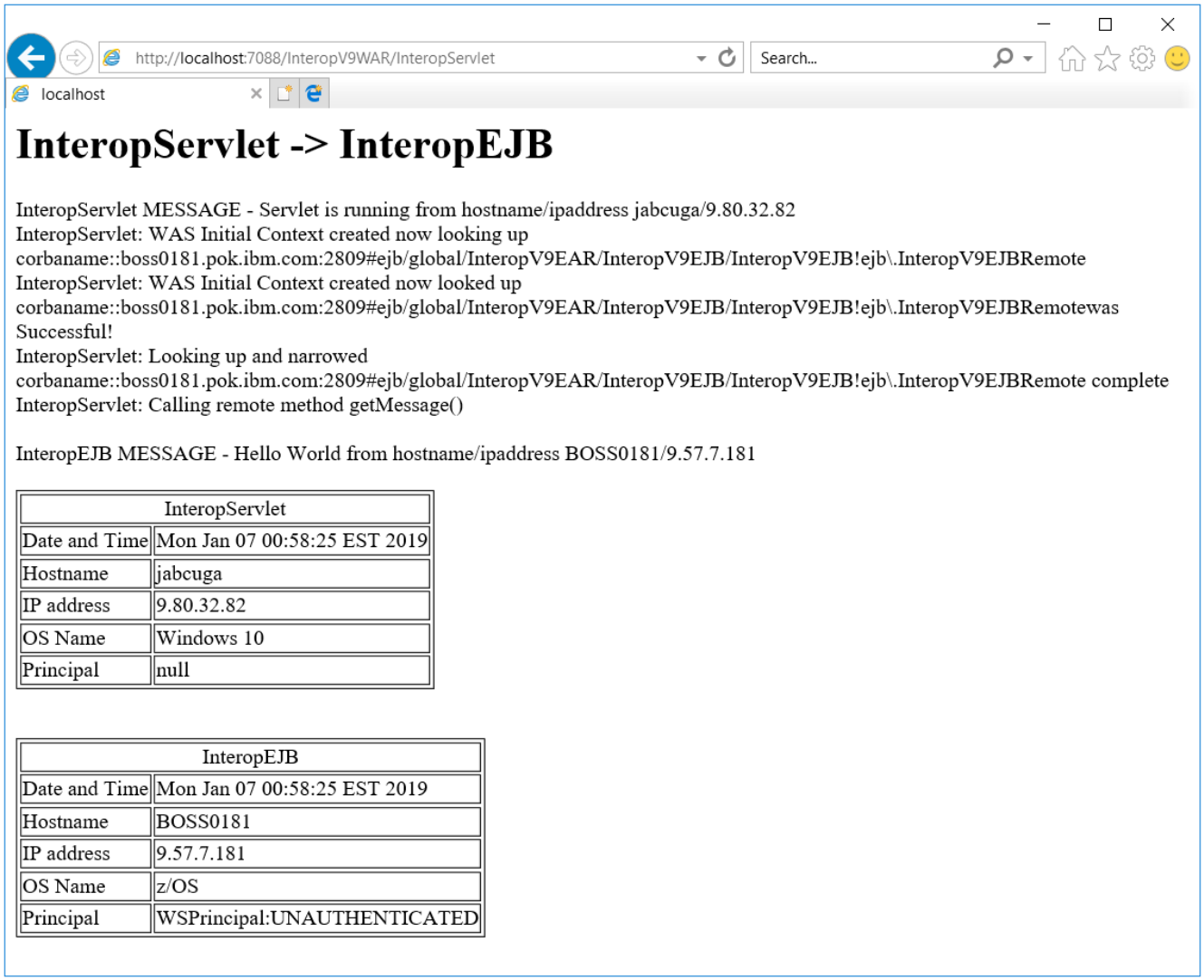
```
<basicRegistry id="basic" realm="BasicRealm">
    <user name="keith" password="keith"/>
    <user name="certID" password="certID"/>
</basicRegistry>

</server>
```

Windows to z/OS – working scenario 1 (unauthenticated)

In this scenario the client is running unauthenticated.



After successful call, the unauthenticated identity is running on the thread in the remote Liberty for z/OS server.



InteropServlet -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

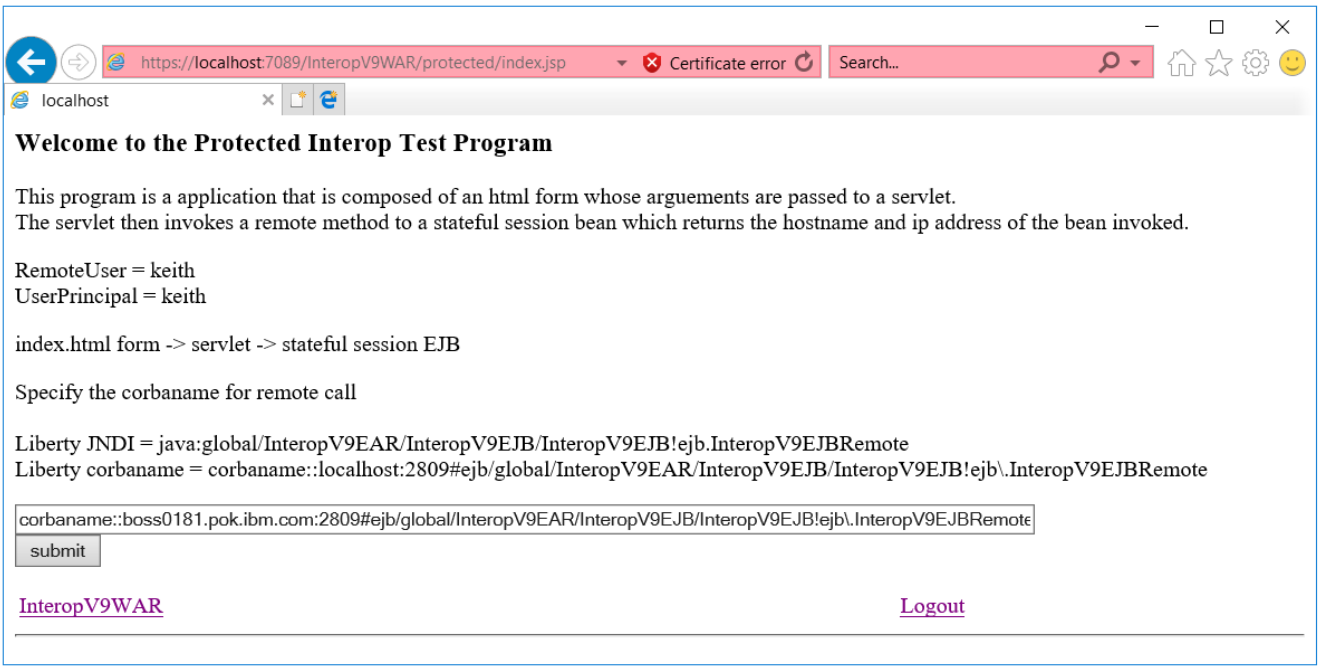
InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

InteropServlet	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	null

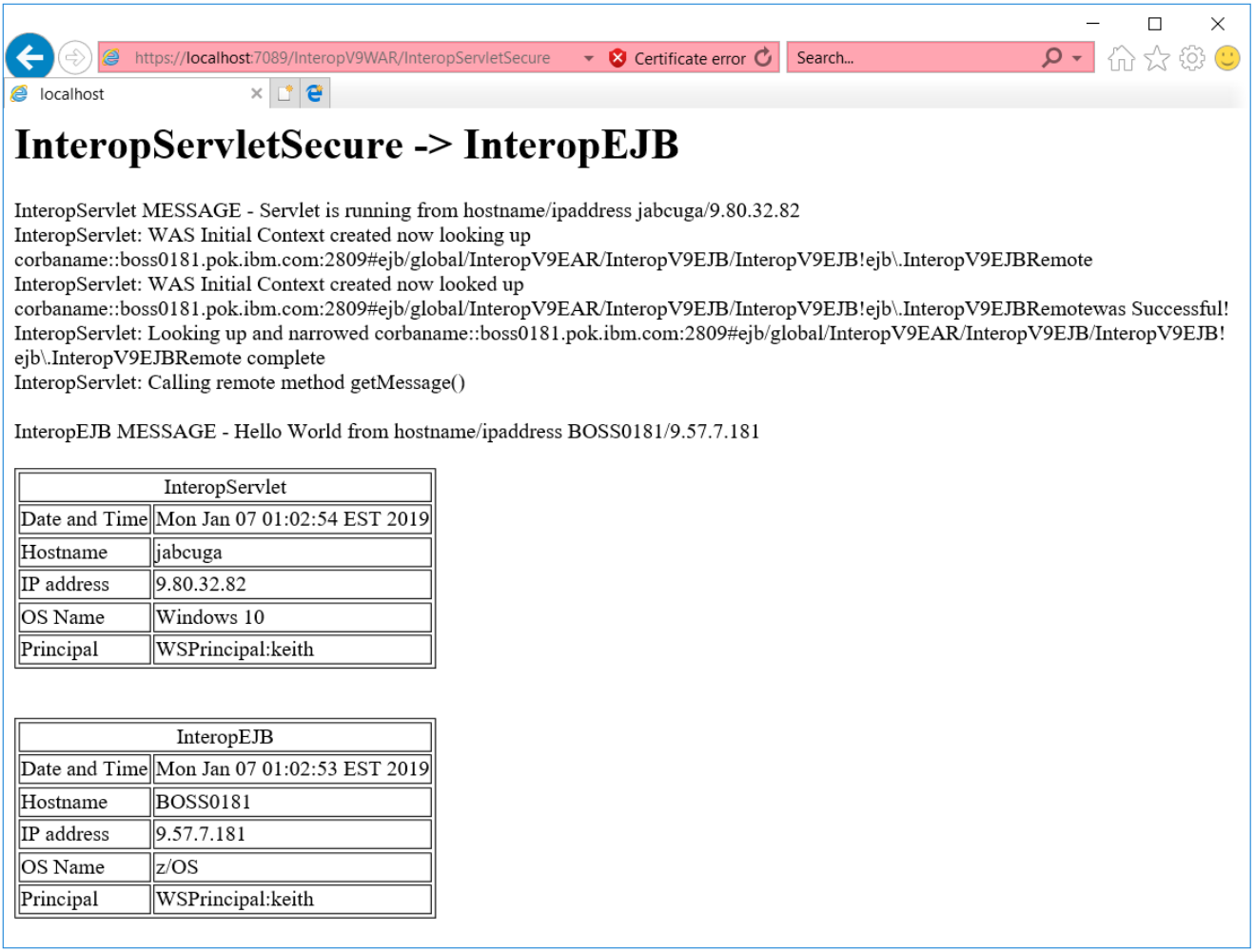
InteropEJB	
Date and Time	Mon Jan 07 00:58:25 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:UNAUTHENTICATED

Windows to z/OS – working scenario 2 (authenticated)

In this scenario the client is authenticated. User “keith” has logged in on the form based web application.



After successful login, the identity “keith” is running on the thread in the remote Liberty for z/OS server.



InteropServletSecure -> InteropEJB

InteropServlet MESSAGE - Servlet is running from hostname/ipaddress jabcuga/9.80.32.82
InteropServlet: WAS Initial Context created now looking up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemote
InteropServlet: WAS Initial Context created now looked up
corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!ejb\.InteropV9EJBRemotewas Successful!
InteropServlet: Looking up and narrowed corbaname::boss0181.pok.ibm.com:2809#ejb/global/InteropV9EAR/InteropV9EJB/InteropV9EJB!
ejb\.InteropV9EJBRemote complete
InteropServlet: Calling remote method getMessage()

InteropEJB MESSAGE - Hello World from hostname/ipaddress BOSS0181/9.57.7.181

InteropServlet	
Date and Time	Mon Jan 07 01:02:54 EST 2019
Hostname	jabcuga
IP address	9.80.32.82
OS Name	Windows 10
Principal	WSPrincipal:keith

InteropEJB	
Date and Time	Mon Jan 07 01:02:53 EST 2019
Hostname	BOSS0181
IP address	9.57.7.181
OS Name	z/OS
Principal	WSPrincipal:keith

Trace

If problems still persist in setting up the InteropV9 application for testing, the following traces can be enabled to help diagnose the issue.

Enable Liberty Trace
Locate your server.xml file:
WLP_OUTPUT_DIR/serverName/server.xml

and add the following line to enable **Liberty SSL** trace.

<logging traceSpecification="*=info:SSL=all"/>

add the following line to enable **Webcontainer, Ejbcontainer and iiop** trace.

<logging traceSpecification="*=info:EJBContainer=all:com.ibm.ws.security.*=all:
com.ibm.ws.transport.iiop.*=all:com.ibm.ws.webcontainer.security.*=all "/>

The Liberty trace will be written to a file called:
WLP_OUTPUT_DIR/serverName/logs/trace.log

Enable JSSE Trace
Create the file jvm.options in the same directory as your server.xml
WLP_OUTPUT_DIR/serverName/jvm.options

Add the line: -Djavax.net.debug=true

The JSSE trace will be written to messages.log and trace.log
WLP_OUTPUT_DIR/serverName/logs/messages.log
WLP_OUTPUT_DIR/serverName/logs/trace.log

It may be easier to read the JSSE trace in messages.log as it won't be interleaved with he Liberty traces.

The JSSE trace requires the server to be restarted.