How-To Library

Informatica

Implementing TLS Security in a PowerExchange Network

Abstract

This article describes how to implement TLS security in an Informatica® PowerExchange® network.

To implement TLS security, configure TLS and PowerExchange components on the client and server machines.

Supported Versions

- PowerExchange 10.2 HotFix 1

Table of Contents

Introduction

You can configure Transport Layer Security (TLS) protocol communication in a PowerExchange network to ensure secure communication between a PowerExchange server and its clients. TLS communication is based on the Secure Socket Layer (SSL) protocol. This article assumes that you have a basic understanding of the TLS protocol.

To configure TLS communication, establish certificates and keys that authorize the secure connection between systems and enable encryption and decryption of data. Each server or client machine has a TLS private key and a TLS certificate. To enable a secured connection, you must perform some configuration tasks in PowerExchange, and you must create and install valid security keys and certificates using a third-party tool such as OpenSSL.

Note: The security configuration for your organization, and therefore the tasks in this article, are the responsibility of your security administrator. Security administration requires specific knowledge of and permissions on the participating systems. For information about obtaining and distributing certificates on your network, refer to your organization's security administrator or internal documentation. Informatica Global Customer Support has limited ability to support the creation or distribution of security certificates within your organization and cannot change your organization's security policies.

After configuration, the TLS handshake sets up the secure connection. The individual data messages are encrypted by using the session key that is encoded and exchanged during the handshake. PowerExchange supports TLS communication for the following operating systems:

- Linux
- UNIX
- Windows
- z/OS

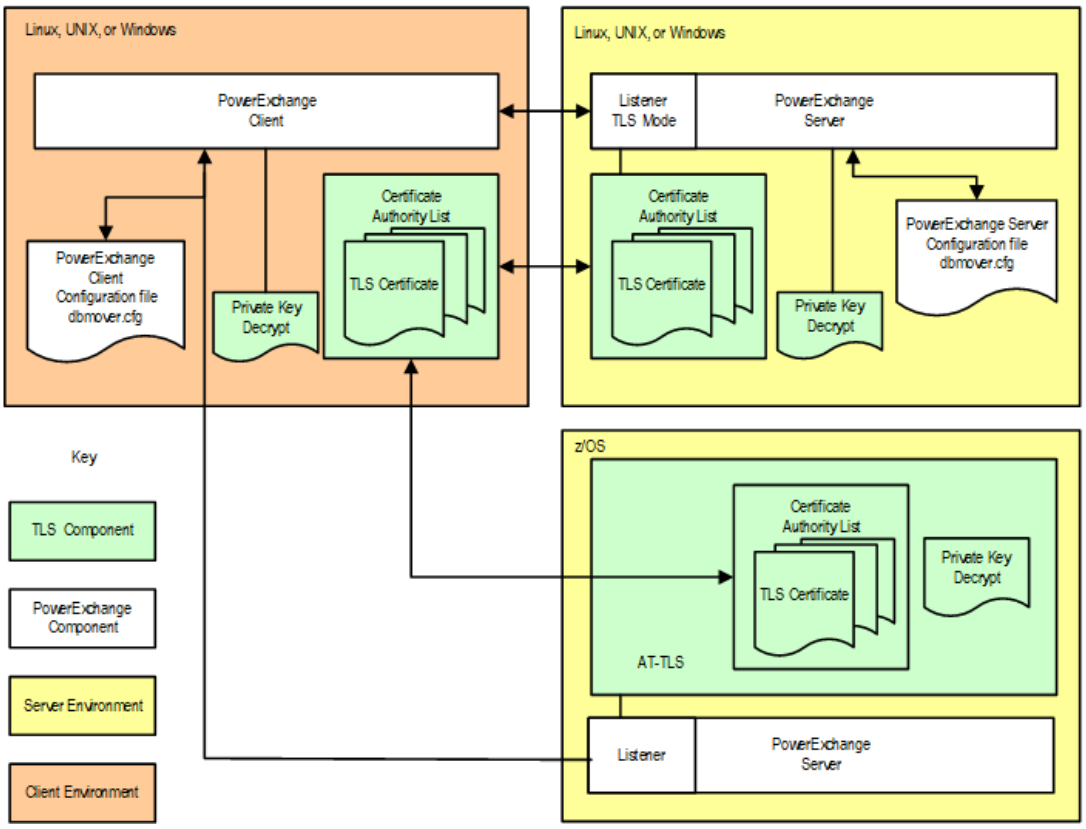
In March 2018, TLS 1.3 was approved by the Internet Engineering Task Force (IETF). This article refers to TLS instead of SSL except when SSL is part of a product or component name.

PowerExchange TLS Architecture

The PowerExchange TLS architecture includes the following components:

- OpenSSL installed on each TLS client and server on Linux, UNIX, or Windows
- PowerExchange TLS configured on each TLS client and server on Linux, UNIX, or Windows
- AT-TLS installed and configured for PowerExchange on each TLS server on z/OS
- PowerExchange Listeners configured on server environments
- X.509 certificates installed on each TLS client and server

The following figure illustrates the PowerExchange TLS architecture:



You can use TLS communication for some, all, or none of the connections in the PowerExchange network.

For example, you might configure connections as follows:

- Configure PowerExchange Listeners to use separate ports for TLS and non-TLS connections.
- Configure the PowerCenter Integration Service client to use TLS connections to PowerExchange Listeners.
- Configure the PowerExchange Navigator and PowerCenter clients to use non-TLS connections to PowerExchange Listeners.

Authentication

You can optionally configure PowerExchange to require server authentication of client certificates, client authentication of server certificates, or both.

If you configure a TLS server to require client authentication, the server requests the client personal certificate with its signing CA certificates. The server checks that the personal certificate of the client is up-to-date and signed by a certificate authority in the CA list of the server. The following statements determine whether the server requires client authentication:

- The SSL_REQ_CLNT_CERT statement in the DBMOVER file of the TLS server on Linux, UNIX, or Windows
- The HandshakeRole statement in the TTLSEnvironmentAction section of the AT-TLS policy file on z/OS

If you configure a TLS client to require server authentication, the client checks that the personal certificate of the server is in-date and signed by a certificate authority in the CA list of the client. The SSL_REQ_SRVR_CERT statement in the DBMOVER file of the TLS client determines whether the client requires server authentication.

If you configure neither the TLS server nor client to require authentication of peer certificates, network packets are still encrypted during the session.

Because clients do not typically require server authentication in a PowerExchange network, you are likely to use one of the following authentication modes:

- Server requests client authentication.
- Neither client nor server requests authentication.

FIPS 140-2 Compliance

TLS, in an appropriate environment, complies with the Federal Information Processing Standard (FIPS) Publication 140-2. By configuring PowerExchange appropriately, creating the necessary certificates, and selecting appropriate algorithms, you can achieve FIPS 140-2 Security Level 1 compliance on a PowerExchange network. To ensure that your PowerExchange network meets the requirements for FIPS 140-2 compliance, consult your security administrator.

FIPS 140-2 Compliance Considerations on z/OS

FIPS 140-2 level 1 support in z/OS System SSL requires z/OS 1.11 or later, or z/OS 1.10 with the fixes for APAR OA26457. Additional fixes and z/OS configuration changes are also required.

In particular, the operating system must include the Cryptographic Services Security Level 3 (FMID JCPT391) component of z/OS System SSL.

z/OS AT-TLS uses z/OS System SSL. For z/OS 1.12 and later, AT-TLS provides the following features in support of FIPS 140-2 compliance:

- A configuration parameter to request that System SSL uses only FIPS 140-2 compliant encryption methods
- Symbolic names for recent encryption methods such as those using AES-256 bit encryption

For earlier version of z/OS, you can enforce the use of FIPS-compliant encryption by specifying the candidate encryption methods as hexadecimal codes rather than as symbolic names.

For more information about achieving FIPS 140-2 Level 1 compliance with System SSL, see the following IBM publications:

- *Cryptographic Services System Secure Sockets Layer Programming*
- *APAR OA26457 System Secure Sockets Layer Programming*

FIPS 140-2 Compliant Cipher Suites

During a TLS handshake, the client and server agree on a symmetric algorithm to use to encrypt data during the session. The client offers a list of cipher suites, and the server selects one from the list. For the PowerExchange network to be FIPS 140-2 compliant, the selected cipher suite must be FIPS 140-2 compliant.

On Linux, UNIX, or Windows clients or servers, PowerExchange uses the OpenSSL runtime engine. When a client and server are both using OpenSSL, the cipher suite that PowerExchange selects is FIPS 140-2 compliant.

On z/OS, AT-TLS manages TLS sessions. The order of cipher suites in the TTLSCipherParms statement in the AT-TLS policy file is important. The server selects the first cipher suite in the list that matches one offered by the client. In this process, ciphers are identified with hexadecimal cipher suite numbers.

To ensure that a z/OS server selects a FIPS 140-2 compliant cipher suite, verify that the first cipher suite in the TTLSCipherParms list matches one of the FIPS 140-2 compliant cipher suites that OpenSSL supports.

The following table is a partial list of FIPS 140-2 compliant cipher suites that OpenSSL and AT-TLS both support:

OpenSSL Cipher Suite Name	AT-TLS Cipher Suite Name	Hexadecimal Value
DHE-RSA-AES256-SHA	TLS_DHE_RSA_WITH_AES_256_CBC_SHA	39
DHE-DSS-AES256-SHA	TLS_DHE_DSS_WITH_AES_256_CBC_SHA	38
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA	35
EDH-RSA-DES-CBC3-SHA	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	16
EDH-DSS-DES-CBC3-SHA	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	13
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	0A
DHE-RSA-AES128-SHA	TLS_DHE_RSA_WITH_AES_128_CBC_SHA	33
DHE-DSS-AES128-SHA	TLS_DHE_DSS_WITH_AES_128_CBC_SHA	32
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA	2F

For a complete list of algorithms that OpenSSL and AT-TLS support, see the OpenSSL and AT-TLS documentation.

Note: The optional, no-charge CPACF feature, which is available on IBM System z machines, provides machine instructions to accelerate hashing algorithms and symmetric key encryption and decryption used with TLS. For performance reasons, you might want to use only those cryptographic suites for which hardware assists are available. For example, CPACF supports AES-128 on z9 and later machines and supports AES-256 on z10 and z196 machines.

PWXUGSK Utility

Use the PWXUGSK utility to generate reports about TLS libraries and certificates that were generated on z/OS for the PowerExchange Listener. You can also determine the validity of certificates that are available to a specified user.

The PWXUGSK utility runs on z/OS and supports the following commands and reports:

- PING command. Use the PING command to verify the following:
  - A specified user ID has the authority to view security certificates for the PowerExchange Listener on z/OS.
  - The certificates available to the PowerExchange Listener are current and valid.
  - The AT-TLS rules defined for the server can intercept inbound requests, remove the TLS information and send TCP/IP packets to the PowerExchange Listener.

  To perform this verification, submit JCL that includes the following command:

  ```
  PWKUGSK CMD=PING PING_LOCATION=node_name
  [PING_UID=user_name {PING_PWD=password|PING_EPWD=encrypted_password}]
  ```
- Certificates report. Reports information about the certificates stored in a RACF keyring or SAF database. To run a certificates report, submit JCL that includes the following command:

  ```
  PWXUGSK CMD=REPORT_CERTIFICATES [LOC_TYPE={KEYRING|DATABASE}] [LOC_NAME=name]
  [DB_PWD=password] [DB_EPWD=encrypted_password] [VERBOSE={N|Y}]
  ```
- Ciphers report. Reports the cipher suites that are available on the z/OS system. To run a ciphers report, submit JCL that includes the following command:

  ```
  PWXUGSK CMD=REPORT_CIPHERS
  ```

- Error codes report. Reports all possible system TLS errors, both from the secure connection and from the processing of data packets. After a TLS failure, particular instances of these errors can be found in the TCP/IP JES message log. To run an error-codes report, submit JCL that includes the following command:

```
PWXUGSK CMD=REPORT_ERROR_CODES
```

Fore more information about the PWXUGSK utility, see the *PowerExchange Utilities Guide*.

PWXUSSL Utility

Use the PowerExchange PWXUSSL utility on Linux, UNIX, or Windows to generate reports about TLS libraries, certificates, and cipher suites on Linux, UNIX, and Windows. You can also convert a PKCS12DER certificate to PEM format, and issue a PING command to a secure server to verify that the server certificates are valid and that the server can transfer TCP/IP packets to a PowerExchange Listener.

For example, to confirm that one of the cipher suites listed in "FIPS 140-2 Compliant Cipher Suites" on page 5 is supported on your machine, generate a report of the cipher suites that are available in the OpenSSL cryptographic library.

The PWXUSSL utility runs from the root PowerExchange installation directory. For example, on a Windows server, you might run the utility from the C:\Informatica\PowerExchange*v.r.m* directory. The PWXUSSL utility supports the following commands and reports:

- CONVERT_PKCS12_PEM command. Converts certificates that were created on z/OS in PKCS12DER format to the PEM format that can be used on Linux, Unix, and Windows machines.
  To convert a certificate, enter the following command:

```
PWXUSSL CMD=CONVERT_PKCS12_PEM INFILE=pkcs12_file_name [PWD=password|
EPWD=encrypted_password] OUT_FILE=pem_file_name [OUT_ENCODING=[DES3|DES_EDE3_CBC|NONE]
OUT_PWD=password]
```

  Note: If OUT_ENCODING is not specified, the DES3 format is used for the output file. If OUT_PASSWORD is not specified, the password or encrypted password associated with the input file is used for the output file.

- PING command. Verifies that a secure connection can be established between the machine from which you issue the command and a PowerExchange Listener on a remote node.
  To PING a connection, enter the following command:

```
PWXUSSL CMD=PING PING_LOCATION=node_name [PING_UID=user_name {PING_PWD=password|
PING_EPWD=encrypted_password}]
```

- Certificate report. Reports information from a certificate chain file.

  To generate a certificate report, enter the following command:

```
PWXUSSL CMD=REPORT_CERTIFICATE INFILE=infile_name [INFORM={PEM|PKCS12|DER}]
[REPORTFORMAT={OPENSSL|SUMMARY|TEXT|ALL}] [PWD=password|EPWD=encrypted_password]
```

- Ciphers report. Reports the cipher suites that are available in the OpenSSL cryptographic library.

  To generate a ciphers report, enter the following command:

```
PWXUSSL CMD=REPORT_CIPHERS [CIPHER_LIST=list] [CONTEXT_METHOD={TLSV1|TLSV1_1|TLSV1_2|
DTLSV1]
```

- Codes report. Reports the return codes from an attempt to establish a secure connection between a PowerExchange Listener and client.
  To generate a codes report, enter the following command:

```
PWXUSSL CMD=REPORT_CODES [CODE_TYPE={ALL|CATYPES|VERIFYRC}]
```

- Configuration report. Reports the the security configuration of the machines that participate in a secured connection. You can filter the results to include a specific participant type, such as a PowerExchange client or PowerExchange Listener.
  To generate a configuration report, enter the following command:

```
PWXUSSL CMD=REPORT_CONFIG [CLIENT_LISTENER_TYPE={ALL|CLIENT|LISTENER}] [NAME=node_name]
```

- Error codes report. Reports error codes returned from SSL processing during an attempt to establish a secure connection between the PowerExchange Listener and client.
  To generate an error codes report, enter the following command:

  ```
  PWXUSSL CMD=REPORT_ERROR_CODES [ERROR_CODE_TYPE={ALL|LIBRARIES|FUNCTIONS|REASONS}]
  ```

- Version report. Reports the version of OpenSSL that was used to build the cryptographic library.

  To generate a version report, enter the following command:

  ```
  PWXUSSL CMD=REPORT_VERSION
  ```

For more information about the PWXUSSL utility, see the *PowerExchange Utilities Guide*.

Implementation Task Flow

Before you begin TLS configuration for secure communication on a PowerExchange network, ensure that your organization has a local CA certificate from a well-known CA vendor. Optionally, you can generate a self-signed certificate for internal uses within your organization's network, such as for testing.

A security administrator typically performs the tasks to configure TLS. Security administrators have specific permissions and system access that allow them to generate and manage security certificates and policy files.

Note: All certificates created for use with PowerExchange must be generated to the X.509 standard. For example, because the PKCS7 format meets the X.509 standard, it can be used to generate the certificates.

1. Determine your organization's requirements for TLS connections.
2. Configure each TLS server on z/OS.
3. Configure each TLS server on Linux, UNIX, or Windows.
4. If your organization requires client validation, configure each TLS client on Linux, UNIX, or Windows.
5. Make the certificates available to the all PowerExchange servers and clients that require authentication.
6. Verify that secure connections can be established between PowerExchange clients and servers.

Step 1. Determine the TLS Requirements for Your PowerExchange Network

Before you implement TLS on a PowerExchange network, determine the security requirements of your organization.

Gather the following information:

- Which connections in the network require TLS security?
- Does your organization require authentication of TLS client or server certificates, and if so, do you want to allow self-signed certificates?
- Do you need to verify which encryption algorithms are used? Do you need to verify that the system meets FIPS 140-2 requirements?
- How are CA and personal certificates issued in your organization?

Step 2. Configure TLS Servers on z/OS

To configure a TLS server on z/OS, perform the following tasks.

Step 2A. Set Up AT-TLS on z/OS

On z/OS release 1.7 and later, AT-TLS uses a Communications Server policy file to determine which sessions use the TLS protocol.

Before you add a rule to the AT-TLS policy file, verify that the file exists and that the policy Agent is running.

Add a rule to this file that defines PowerExchange Listener properties for TLS communication.

To add a rule, edit the policy file or use the IBM Configuration Assistant for z/OS Communications Server. You can download the IBM Configuration Assistant for z/OS Communications Server from the IBM z/OS Support web site.

When you add a rule, include the following statements:

Statement	Value
LocalPortRange	PowerExchange Listener port number.
Jobname	PowerExchange Listener job name.
Direction	Direction of communication. Specify **Inbound** to indicate that communication proceeds from client to Listener.
TTLSGroupActionRef	References an existing *group_action* that is defined in another section of the policy file.
TTLSEnvironmentActionRef *environment_action*	References an existing *environment_action* that is defined in another section of the policy file.

The following example rule demonstrates how to enter the statements:

```
 TTLSRULE JOB_JBBV861
{
 LocalPortRange           13132
 Jobname                  JBBV861
 Direction                Inbound
 TTLSGroupActionRef       gActEnableTTLS
 TTLSEnvironmentActionRef eActServerDefault
}
```

The TTLSGroupActionRef and TTLSEnvironment ActionRef statements in the rule reference statements in other sections of the policy file. The following table describes the statements that are referenced:

Statement	Sub-Statement	Value
TTLS Group Action	TTLSEnabled	On
	CtraceClearText	Off
	Trace	7

Statement	Sub-Statement	Value
TTLSEnvironmentAction	HandshakeRole	For servers, specifies one of the following values: - **Server**. The Listener acts as the TLS server and does not require client authentication. Little or no peer subject certificate verification is performed. Use this mode to establish a quick configuration with only network encryption. - **ServerWithClientAuth**. The Listener acts as the TLS server and requires client authentication. This mode requires AT-TLS to verify that the subject certificate is issued from a remote Linux, Unix, or Windows client with a CA certificate that the z/OS system trusts. This mode provides more security, and requires that a security administrator with knowledge of certificate administration coordinate the certificates on both the z/OS and Linux, Unix, or Windows servers.
	TTLSCipherParmsRef	References the TTLSCipherParms statement.
	TTLSKeyRingParmsRef	References the TTLSKeyRingParms statement.
TTLSCipherParms	V3CipherSuites	Supported symmetric cipher suites.
TTLSKeyRingParms	Keyring	Key ring that contains the personal and CA certificates.

The following example statements are referenced by the rule in the policy file:

```
 TTLSGroupAction   gActEnableTTLS
{
 TTLSEnabled            On
 CtraceClearText        Off
 Trace                  7
}
 TTLSEnvironmentAction  eActServerDefault
 {
  HandshakeRole         Server
  TTLSCipherParmsRef    cipher1~AT-TLS__Silver
  TTLSKeyringParmsRef   kATTLSkeyring
}
 TTLSCipherParms        cipher1~AT-TLS__Silver
{
 V3CipherSuites         TLS_RSA_WITH_DES_CBC_SHA
 V3CipherSuites         TLS_RSA_WITH_3DES_EDE_CBC_SHA
 V3CipherSuites         TLS_RSA_WITH_AES_128_CBC_SHA
}
 TTLSKeyRingParms       kATTLSkeyring
 {
 Keyring                ATTLS_keyring
 }
```

Step 2B. Create Personal Certificates on z/OS

To create a personal certificate on z/OS, use RACF or another security facility such as ACF2 or TopSecret.

The following steps assume that you are using RACF and the RACF administrator has already created a suitable CA site certificate.

1. Create a key ring.
2. Create a personal certificate.
3. Connect the personal certificate to the key ring.
4. Connect the CA certificate to the key ring.

The following example RACF commands perform these steps:

```
/* Create a Keyring for the application                    */
RACDCERT ID(MYUSERID) ADDRING(ATTLS_keyring)
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

/* Create a certificate for the Server application        */
RACDCERT ID(MYUSERID) GENCERT -
  SUBJECTSDN ( -
    O('MyCompany') -
    CN('MYUSERID.mymachine.myorganization.com') -
    OU('myorganizationunit') -
    C('GB') -
            )          -
   WITHLABEL('MYUSERIDCert1')-
   SIGNWITH(CERTAUTH LABEL('LOCALCA'))
 SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

/* Connect the server certificate to the server's keyring.*/
RACDCERT ID(MYUSERID) CONNECT(ID(MYUSERID) -  LABEL('MYUSERIDCert1') -
RING(ATTLS_keyring) -
DEFAULT -
USAGE(personal))
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH

/* Connect the CA certificate to the server's keyring     */
RACDCERT ID(MYUSERID) CONNECT(CERTAUTH -
LABEL('LOCALCA') -
RING(ATTLS_keyring) -  USAGE(certauth))
SETROPTS RACLIST(DIGTCERT,DIGTNMAP) REFRESH
```

Step 2C. Configure the DBMOVER File on the z/OS TLS Server

In the DBMOVER configuration file, include a LISTENER statement with the port that you associated with the PowerExchange Listener task in the AT-TLS policy file. On z/OS, the LISTENER statement does not include any additional parameters for TLS processing.

For example, you might run the PowerExchange Listener in non-TLS mode on port 13131 to connect to the PowerExchange Navigator and PowerCenter Developer and run the Listener in TLS mode on port 13132 to connect to the PowerCenter Integration Service. In this case, the AT-TLS policy file includes a rule for the Listener job that uses TLS on port 13132, and the DBMOVER file includes the following statements:

```
LISTENER=(node1,TCPIP,13131)
LISTENER=(node1,TCPIP,13132)
```

Caution: Because PowerExchange uses AT-TLS to implement TLS on z/OS, do not include the SSL parameter in the LISTENER statement in the DBMOVER file on z/OS. Also, do not include SSL-related statements, such as SSL_REQ_CLNT_CERT, in the file. Including SSL-related parameters or statements in the DBMOVER file cause processing to fail.

Step 3. Configure TLS Servers on Linux, UNIX, or Windows

To configure a TLS server on Linux, UNIX, or Windows, perform the following tasks.

Step 3A. Set Up OpenSSL on Linux, UNIX, or Windows

OpenSSL must be installed on the Linux, UNIX, or Windows machine on which you create or convert certificates for the TLS server.

OpenSSL is pre-installed on Linux and UNIX.

To install OpenSSL on Windows, download it from http://www.openssl.org.

To verify that OpenSSL is installed properly, launch OpenSSL and issue the following command:

```
OpenSSL> version
```

OpenSSL displays the version number. PowerExchange works with any version of OpenSSL.

Step 3B. Create CA Certificates on Linux, UNIX, or Windows

Create a CA certificate that you can use to sign personal certificates on Linux, UNIX, or Windows.

If you already have a CA certificate that you can use to sign personal certificates, skip this step.

1. At the command prompt, enter the following command:

   ```
   openssl
   ```

   The OpenSSL> prompt appears.

2. To generate a private key and a request for a CA certificate, issue the OpenSSL req command:

   ```
   OpenSSL> req -newkey rsa:2048 -digest -keyout rootkey.pem -out rootreq.pem
   ```

   Options:

   -newkey rsa:2048

   Requests a new certificate request and a 2048-bit RSA private key.

   -digest

   Specifies the message digest used to sign the request. A value specified in this option overrides the message digest specified in the configuration file, unless a public key algorithm is configured that overrides this choice. For example, a DSA signature will always use an SHA1 digest. The generic name **dgst** can also be specified. The default digest is SHA256. To see a list of supported algorithms, use the list --digest-commands command.

   -keyout *rootkey*.pem

   Specifies the name of a file to which the private key for the CA certificate is written.

   -out *rootreq*.pem

   Specifies the name of a file to which the certificate request for the CA certificate is written.

3. Respond to the series of prompts that OpenSSL displays.

   For most prompts, you can accept the default. For Common Name, you can use the server name.

4. To generate a public CA certificate, issue the OpenSSL x509 command:

   ```
   OpenSSL> x509 -req -in rootreq.pem -digest -extensions V3_CA -signkey rootkey.pem -out
   rootcert.pem -days 999
   ```

   Options:

   -req

   Specifies that the input is a certificate request rather than a certificate.

   -in *rootreq*.pem

   Specifies the name of the input certificate request file that you created in step 2.

   -digest

   Specifies the message digest used to sign the request. A value specified in this option overrides the message digest specified in the configuration file, unless a public key algorithm is configured that overrides this choice. For example, a DSA signature will always use an SHA1 digest. The generic name **dgst** can also be specified. The default digest is SHA256. To see a list of supported algorithms, use the list --digest-commands command.

-extensions V3_CA

Specifies V3_CA as the section from which to add certificate extensions. Use this option to convert a certificate request into a self-signed certificate by using extensions for a CA.

-signkey *rootkey*.pem

Specifies use of the private key that you created in step 2 to sign this certificate.

-out *rootcert*.pem

Specifies the name of the file to which the CA certificate is written. If you require authentication, specify this value for the CALIST parameter of the SSL statement in the DBMOVER file.

-days

Specifies the number of days before the certificate expires, from 1 to 999. Default is 30.

Step 3C. Create Personal Certificates on Linux, UNIX or Windows

To create a personal certificate, issue the OpenSSL req and x509 commands and then concatenate the two files that these commands create.

Alternatively, you can create a personal certificate by using one of the following methods:

- Export an existing Windows certificate, and convert it to .pem format by using OpenSSL.
- Generate a personal certificate on the z/OS system by using RACF and export the certificate to the Linux, UNIX, or Windows system. Then convert the certificate to the .pem format by using OpenSSL or the PWXUSSL utility.

1. If the OpenSSL program is not running, enter the following command at the command prompt to start the program:

   ```
   openssl
   ```

   The OpenSSL> prompt appears.
2. To generate a private key and a request for a personal certificate, issue the following OpenSSL req command:

   ```
   OpenSSL> req -newkey rsa:2048 -digest -keyout personalkey.pem -out personalreq.pem
   ```

   Options:

   -newkey rsa:2048

   Requests a new certificate request and a 2048-bit RSA private key.

   -digest

   Specifies the message digest used to sign the request. A value specified in this option overrides the message digest specified in the configuration file, unless a public key algorithm is configured that overrides this choice. For example, a DSA signature will always use an SHA1 digest. The generic name **dgst** can also be specified. The default digest is SHA256. To see a list of supported algorithms, use the list --digest-commands command.

   -keyout *personalkey*.pem

   Specifies the name of a file to which the private key for the personal certificate is written.

   -out *rootreq*.pem

   Specifies the name of a file to which the certificate request for the personal certificate is written.
3. At the prompt for a passphrase, enter the passphrase that you provided when you created the CA certificate.
4. At the prompt for a PEM passphrase, enter a second passphrase. You will specify this passphrase in the PASS parameter of the SSL statement in the DBMOVER configuration file.
5. Respond to the series of prompts that OpenSSL displays. You can use the same responses that you provided for the CA certificate request.

6. To generate a personal certificate, issue the following OpenSSL x509 command:

```
OpenSSL> x509 -req -in personalreq.pem -digest -CA rootcert.pem -CAkey rootkey.pem -
CAcreateserial -out personalcert.pem -days 999
```

Options:

-req

Specifies that the input is a certificate request rather than a certificate.

-in *personalreq*.pem

Specifies the name of the input certificate request file that you created in step 2.

-digest

Specifies the message digest used to sign the request. A value specified in this option overrides the message digest specified in the configuration file, unless a public key algorithm is configured that overrides this choice. For example, a DSA signature will always use an SHA1 digest. The generic name **dgst** can also be specified. The default digest is SHA256. To see a list of supported algorithms, use the list --digest-commands command.

-CA *rootcert*.pem

Specifies the name of the file that contains the CA root certificate that was created in the step 2.

-CAkey *rootkey*.pem

Specifies the use of the CA private key that was created in the step 2 and is that is used to sign this certificate.

CAcreateserial

Creates the CA serial number file.

-out *personalcert*.pem

Specifies the name of the file to which the personal certificate is written.

-days

Specifies the number of days before the certificate expires, from 1 to 999. Default is 30.

7. Concatenate the personal key and personal certificate.

On Windows, enter the following command:

```
type personalcert.pem personalkey.pem > personalcertkey.pem
```

On Linux or UNIX, enter the following command:

```
cat personalcert.pem personalkey.pem > personalcertkey.pem
```

This step creates the file *personalcertkey*.pem. Specify this value in the KEY parameter of the SSL statement in the DBMOVER configuration file.

Step 3D. Configure the DBMOVER File on the Linux, UNIX, or Windows TLS Server

Depending on how you want to configure TLS processing, you must include some or all of the following statements in the DBMOVER file on the Linux, UNIX, or Windows TLS server:

- LISTENER. Specifies the parameters for the PowerExchange Listener that is operating in TLS mode.
- SSL. Specifies the security key and passphrase that you are using to make the TLS connection. If the server requests client authentication, also specifies the certificate authority list (CALIST).

- SSL_REQ_CLNT_CERT. Specifies whether the OpenSSL library verifies that the issuer of the remote server subject certificate is a locally trusted CA certificate specified in the SSL CALIST or CAPATH parameter. Set this parameter to Y if the following conditions are true:
  - You need to verify the identify of the remote machine. For example, another machine might be impersonating a client on your network.
  - You have or your security administrator has a technical understanding of certificates and can coordinate the locally trusted CA certificates with the remote client subject certificates.
  - You have a PowerExchange process that runs on Linux, Unix or Windows. For example, you are accessing Oracle, Microsoft SQL Server, DB2 UDB or MySQL data.
- SSL_ALLOW_SELFSIGNED. Specifies whether to allow self-signed certificates if the server requests client authentication. Set this statement to Y if certificate verification is performed, and the OpenSSL library rejects the local CA certificate. If SSL_REQ_CLIENT_CERT is set to Y, the OpenSSL library rejects self-signed CA certificates by default. A CA certificate is considered trusted if it is present in the location specified by the SSL CALIST or CAPATH parameters.
  In general, only a single CA certificate is required. However, if a chain of certificates is included, the bottom certificate may be self-signed.

  Note: If the local CA certificate is rejected once, it will also fail for all other incoming connection attempts.

The following DBMOVER statements configure TLS communication:

```
LISTENER=(node,TCPIP,port_number,,,,,,,SSL)
SSL=(PASS=passphrase,KEY=personalkey.pem),CALIST=calist
SSL_REQ_CLNT_CERT=Y
```

You might want to run the Listener in both TLS and non-TLS mode. For example, you could run the Listener in non-TLS mode on port 13131 to connect to the PowerExchange Navigator and PowerCenter Developer, and run the Listener in TLS mode on port 13132 to connect to the PowerCenter Integration Service. In this case, the DBMOVER file would include the following LISTENER statements:

```
LISTENER=(node1,TCPIP,13131)
LISTENER=(node1,TCPIP,13132,,,,,,,SSL)
```

Step 4. Configure TLS Clients on Linux, UNIX, or Windows

To configure a TLS client on Linux, UNIX, or Windows, perform the same tasks for each client as you did for the Linux, UNIX, or Windows server.

Step 4A. Set up OpenSSL

Step 4A. Set Up OpenSSL. See "Step 3A. Set Up OpenSSL on Linux, UNIX, or Windows" on page 11 for more information.

Step 4B. Create a CA Certificate

Step 4B. Create a CA certificate. See "Step 3B. Create CA Certificates on Linux, UNIX, or Windows" on page 12 for more information.

Step 4C. Create a Personal Certificate

Step 4C. Create a personal certificate. See "Step 3C. Create Personal Certificates on Linux, UNIX or Windows" on page 13 for more information.

Step 4D. Configure TLS Clients on Linux, UNIX or Windows

Depending on how you want to configure TLS processing, you must include some or all of the following statements in the DBMOVER file on the Linux, UNIX, or Windows client machine:

- NODE. Specifies the server to which to connect in TLS mode. In this statement, include the following parameters:
  - Specify the ZOSSSL parameter when accessing a z/OS server in TLS mode.
  - Specify the SSL parameter to access a Linux, UNIX, or Windows server in TLS mode.
  - Specify N as the last parameter of the NODE statement if the remote peer does not verify certificates. For example, if the AT-TLS HANDSHAKE ROLE is set to SERVER, or if the SSL_REQ_CLNT_CERT option is set to N, then specify N as the last parameter in the NODE statement.
- SSL. Specifies the TLS key and passphrase that you use to make the connection. If the client requests server authentication, also specifies the certificate authority list (CALIST).
- SSL_REQ_SRVR_CERT. Specifies whether the OpenSSL library verifies that the issuer of the remote server subject certificate is a locally trusted CA certificate specified in the SSL CALIST or CAPATH parameter. Set this parameter to Y if both of the following conditions are true:
  - You need to verify the identify of the remote machine. For example, another machine might be impersonating a server on your network.
  - You have or your security administrator has a technical understanding of certificates and can coordinate the locally trusted CA certificates with the remote PowerExchange Listener subject certificates.
- SSL_ALLOW_SELFSIGNED. Specifies whether to allow self-signed certificates if the client requests server authentication. Set this statement to Y if certificate verification is performed, and the OpenSSL library rejects the local CA certificate.

  Note: If the local CA certificate is rejected once, it will also fail for all other incoming connection attempts.

For example, the following DBMOVER statements configure TLS communication and disable the authentication of peer certificates:

```
NODE=(server_listener,TCPIP,remote_host,port_number,,,,,SSL)
SSL=(PASS=passphrase,KEY=personalkey.pem)
SSL_REQ_CLNT_CERT=N
SSL_REQ_SRVR_CERT=N
```

When you disable authentication of peer certificates, you can omit the CALIST and CAPATH parameters from the SSL statement.

Step 5. Make Certificates Available

Make CA certificates available to any TLS client or server that requires authentication of peer certificates.

To make CA certficates available, perform the following actions:

- Copy the certificates to the client or server machine that requires authentication of peer certificates.
- Install the certificates using an appropriate program, such as OpenSSL.
- In the DBMOVER file, update the CALIST or CAPATH parameter in the SSL statement to point to the CA certificates.

Step 6. Test the Secure Connection between the Client and the Server

To verify that a secure connection can be established between the PowerExchange client and server, ping the remote PowerExchange Listener.

On Linux, UNIX, or Windows use the PWXUSSL utility to issue a PING command:

```
C:\Informatica\PowerExchangev.r.m pwxussl CMD=ping LOCATION=node_name PING_UID=user_name
PING_PWD=password
```

On z/OS, use either the PWXUGSK utility or the DTLREXE utility to issue a PING command:

```
//GSKPINGN JOB 'GSK      ',MSGLEVEL=(1,1),MSGCLASS=X,
//       CLASS=A,NOTIFY=&SYSUID
//*
//*============================================================
//* RUN PWXUGSK
//*============================================================
//STEP1   EXEC PGM=PWXUGSK,REGION=900M,
//          PARM='CMD=PING PING_LOCATION=ZSY216495
//          PING_UID=MY PING_PWD=MYPWD VERBOSE=N'
//*
//STEPLIB  DD DISP=SHR,DSN=&HLQ..LOADLIB
//       DD DISP=SHR,DSN=&SCERUN
//       DD DISP=SHR,DSN=&RUNLIB
//*
//DTLMSG  DD DISP=SHR,DSN=&HLQ..DTLMSG
//DTLKEY DD DISP=SHR,DSN=&RUNLIB(LICENSE)
//DTLSGN DD DISP=SHR,DSN=&RUNLIB(SIGNON)
//DTLLOG   DD DUMMY,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133)
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
//DTLCFG  DD *
/*
```

or

```
DTLREXE PROG=PING LOC=node
```

The following example shows a sample report returned by a PING command to indicate whether a secure connection was successfully established:

```
PING to location 'node_name'
============================
rc = 0 from OS_Get_Module_EP() on SSLOPNWP
rc = 0 from SSW_Init()
rc=0 from SSW_Env_Open()

Creating TCPIP socket connection to ipaddr 'node' portno 16495
------------------------------------------------------------------
rc=0 from WSAStartup()
rc=0 from getaddrinfo() ipaddr=node portno=16495
socket number 380 returned from socket()
rc=0 from connect() socket=380

Creating OpenSSL secure socket connection to ipaddr 'node' portno 16495
-------------------------------------------------------------------------
rc=0 from SSW_Socket_Open()
Secure socket created

Sending the Connect request
---------------------------
PacketSize 1344 = 2 * 96 (HDR) + 2 * 576 (LST_HDR_LEN)
rc=0 from SSW_Send()
Sent a Connect request. packet len=1344 WrittenLen=1344
Calling SSW_Receive() for 96 bytes
```

```
rc=0 from SSW_Receive(). Received 96 bytes

Checking if the socket is readable
----------------------------------
Socket 380 is not readable as expected.
The PowerExchange listener has accepted the connection.

Sending the DTLREXEL PING request
---------------------------------
PacketSize 111 = 96 (HDR) + 15 (Data length)
rc=0 from SSW_Send()
Sent a DTLREXEL request. packet len=111 WrittenLen=111
Calling SSW_Receive() for 96 bytes
rc=0 from SSW_Receive(). Received 96 bytes

Terminating
-----------
rc=0 from SSW_Env_Close()
rc=0 from closesocket() on socket 380
rc=0 from OS_Unload_Module() on SSLOPNWP

Ping has run successfully at the TCPIP / SSL level.
Check displays in remote listener log.
```

Terms and Acronyms

The following terms and acronyms are used in this article:

Application-Transparent Transport Layer Security (AT-TLS)

An application that transparently implements z/OS System SSL in the TCP layer of the TCP/IP stack.

certificate authority (CA)

An organization that issues certificates and vouches for the identities of the subjects of the certificates.

certificate chain

A series of certificates, which includes a personal certificate, the certificate for the root authority, and any intermediate certificate authorities.

Federal Information Processing Standard Publication 140-2 (FIPS 140-2)

A U.S. government computer security standard used to accredit cryptographic modules.

OpenSSL

An open source implementation of the SSL and TLS protocols for Linux, UNIX, and Windows types of operating systems.

personal certificate

A public key certificate that identifies a subject and the subject's public key. The certificate is digitally signed by a CA. On a PowerExchange network, each SSL client and server must have a personal certificate installed.

root authority

Master certificate authority, the highest level in a hierarchy of certificate authorities.

Secure Socket Layer (SSL)

A cryptographic protocol that provides security for communications over networks. Predecessor of the TLS protocol.

self-signed certificate

A certificate that is signed by its own creator.

site certificate

A certificate that can be used to sign other certificates and functions as a CA certificate in a RACF environment.

System SSL, z/OS System SSL

z/OS implementation of the SSL and TLS protocols.

Transport Layer Security (TLS)

A cryptographic protocol that provides security for communications over networks. TLS is the successor to the Internet Engineering Task Force (IETF) standard version of the SSL protocol.

X.509

An ITU-T standard for a public key infrstructure. X.509 specifies standard formats for public key certificates.

Authors

John Boyle

Ross Ferrand

Barbara Green