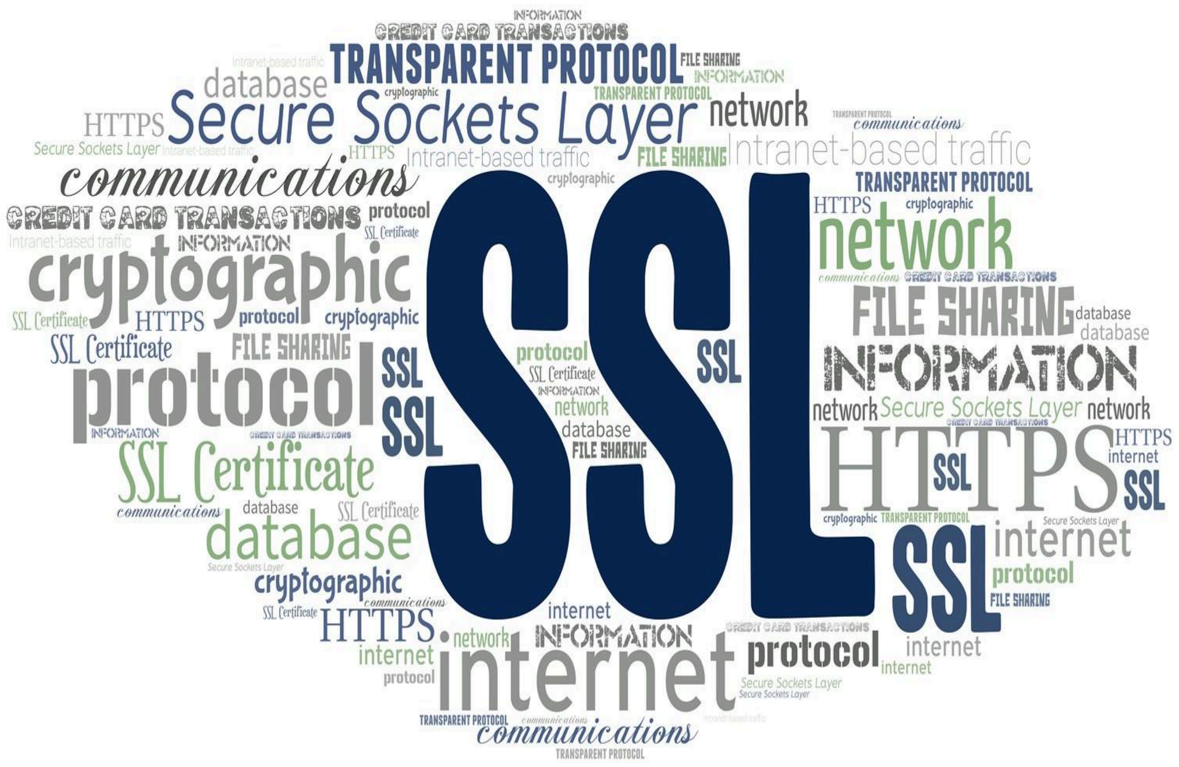


Understanding and Configuring SSL for Reliable and Secure Integrations in IBM App Connect Enterprise

A Guide to SSL for Secure App Connect Integrations

-Votarikari Shravan(912)

Table of Contents

Overview

Secure Sockets Layer (SSL) is a cryptographic protocol that ensures secure communication over a computer network. It works by encrypting data transmitted between two parties, preventing unauthorized access and ensuring data integrity.

What it does:

- Encryption: Creates a secure tunnel for data transfer by scrambling information using cryptographic algorithms. This makes it unreadable for anyone without the appropriate decryption key.
- Authentication: Verifies the identity of both communicating parties, ensuring you're interacting with the intended recipient and not an impersonator.
- Data integrity: Protects data from alteration during transmission, guaranteeing that the information received hasn't been tampered with.

Applications:

- HTTPS: Encrypts communication between web browsers and servers, protecting login credentials, financial information, and other sensitive data.
- Email: Secures email exchange by encrypting messages and attachments.
- Messaging apps: Provides secure communication channels for instant messaging applications.
- Virtual Private Networks (VPNs): Creates a secure tunnel for data transmission over public networks.

Benefits:

- Enhanced security: Protects against eavesdropping, data breaches, and man-in-the-middle attacks.
- Increased trust and confidence: Enables secure online transactions and interactions.
- Improved privacy: Keeps sensitive information confidential.

Evolution and future:

- SSL has been superseded by Transport Layer Security (TLS). Both protocols share similar functionalities, but TLS offers stronger encryption algorithms and improved security mechanisms.
- TLS continues to evolve with new versions and features. Staying updated with the latest TLS versions is crucial for maintaining a strong security posture.

different methods of verifying the identities of the parties involved in communication

1. One-Way SSL Authentication
2. Two-Way SSL Authentication

> *Depending upon which party authenticated they are involved in communication, categorized into two types. If only the server authenticates itself to the client that is called One-way ssl authentication. If both server and client authenticates mutually that is called Two-way authentication.*

One-way SSL Authentication:

- **Server Authentication:** Only the server authenticates itself to the client using a digital certificate issued by a trusted Certificate Authority (CA).
- **Client Trust:** The client trusts the server's certificate because it is issued by a trusted CA. It does not actively verify the client's identity.
- **Use Cases:** Websites where data primarily flows from server to client (e.g., news websites, static content delivery), situations where client identity is not crucial.

Two-way SSL Authentication (Mutual Authentication):

- **Mutual Verification:** Both the server and the client authenticate themselves to each other using digital certificates.
- **Enhanced Security:** Provides stronger security as both parties are verified, reducing risk of impersonation and man-in-the-middle attacks.
- **Use Cases:** Online banking, financial transactions, secure login portals, applications where sensitive data exchange occurs.

Choose either One-way or Two-way Authentication

- One-way authentication is typically used when security requirements are less stringent and only server-to-client communication needs to be secured.
- For scenarios requiring mutual authentication, where both parties verify each other's certificates, both client and server would need to configure their respective trust stores.
- Always implement proper certificate management practices to ensure trust stores contain valid and updated certificates for secure communication.

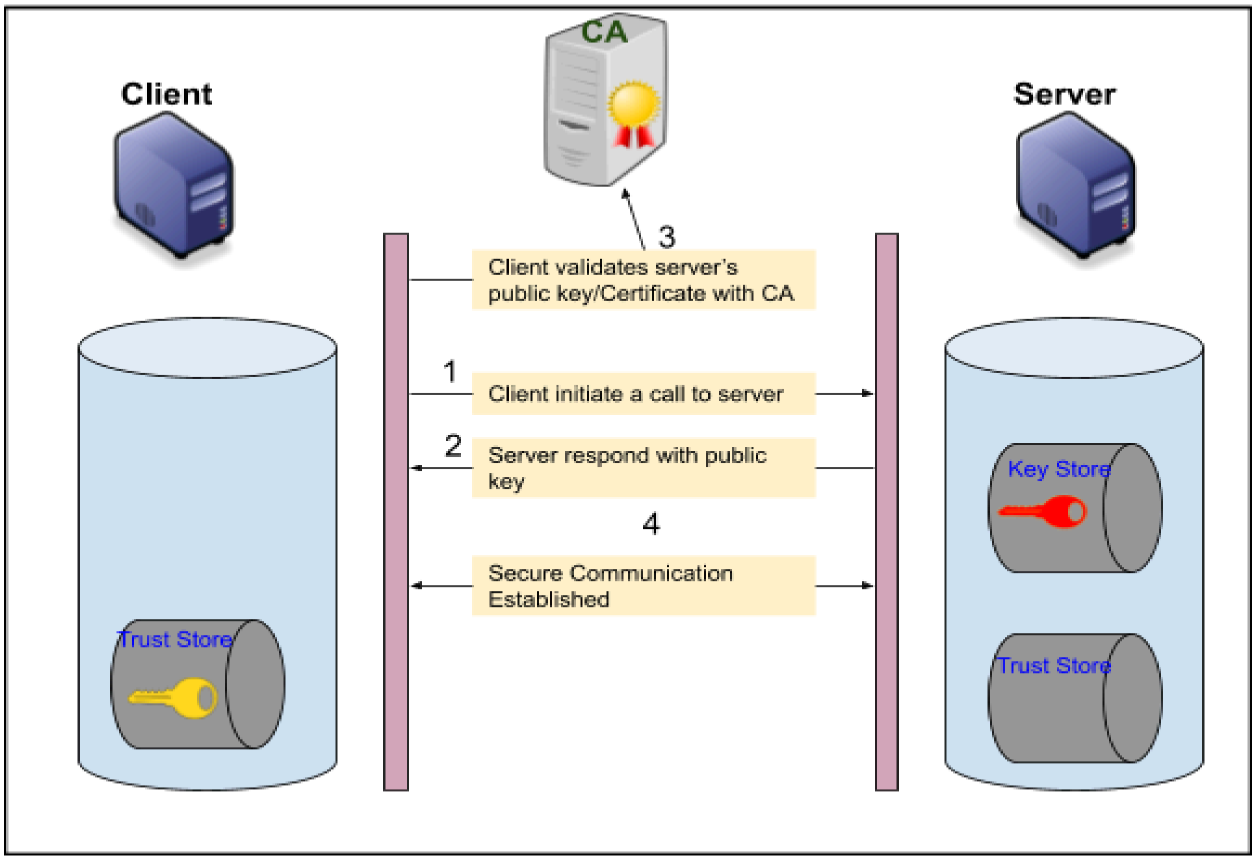
Key entities play crucial roles in authentication

1. **Client:** The individual or system initiating the communication with the server. In web browsing, it's your web browser accessing a website.
2. **Server:** The system hosting the data or service the client wants to access. For websites, it's the web server where the website files reside.
3. **Certificate Authority (CA):** A trusted third-party organization responsible for verifying the identity of the server and issuing its digital certificate. Verisign, DigiCert, and GlobalSign are some recognized CAs.
4. **Public Key Infrastructure (PKI):** The network of CAs and trusted certificates that enables secure authentication and encryption. Think of it as a system of digital identities and verification mechanisms.
5. **Server Certificate:** A digital document issued by the CA containing information about the server, including its domain name, organization, and public key. This certificate serves as the server's proof of identity.
6. **Client Trust Store:** A database within the client software (e.g., web browser) storing trusted CA certificates. When verifying a server certificate, the client checks if the issuer CA is present in its trust store, ensuring its legitimacy.
7. **Secure Sockets Layer (SSL) or Transport Layer Security (TLS):** The cryptographic protocols used to establish a secure connection between the client and server. One-way

authentication primarily relies on the TLS handshake process, involving encryption and key exchange mechanisms.
8. **Cipher suites:** Algorithms used for encryption and decryption during the TLS handshake and data transmission. Choosing appropriate cipher suites ensures strong and secure communication channels.
9.**KeyDatabase :** A key database is a file that the client or server uses to store one or more key pairs and certificates.

Steps involved during One-way Authentication

1. Client initiates communication with the server.
2. Server sends its digital certificate to the client.
3. Client checks the server certificate's signature against the issuer CA's certificate in its trust store.
4. If the verification is successful, the client trusts the server's identity and establishes a secure connection using TLS.
5. Data is then encrypted and transmitted securely between the client and server.


CA
Client
Server
3
Client validates server's public key/Certificate with CA
1
Client initiate a call to server
2
Server respond with public key
4
Secure Communication Established
Key Store
Trust Store
Trust Store

- *If the certificates are CA-Signed Certificates then clients don't have to configure the truststore as the certificate will get validated in the default JRE truststore.*
- *If the certificates are a Self-Signed Certificate then the client needs to create a custom truststore and import the server certificate into that truststore.*

Keystore and truststore file extensions

Keystore and truststore file extensions vary depending on the software and platform used. Here's a summary of some common extensions:

Keystore:

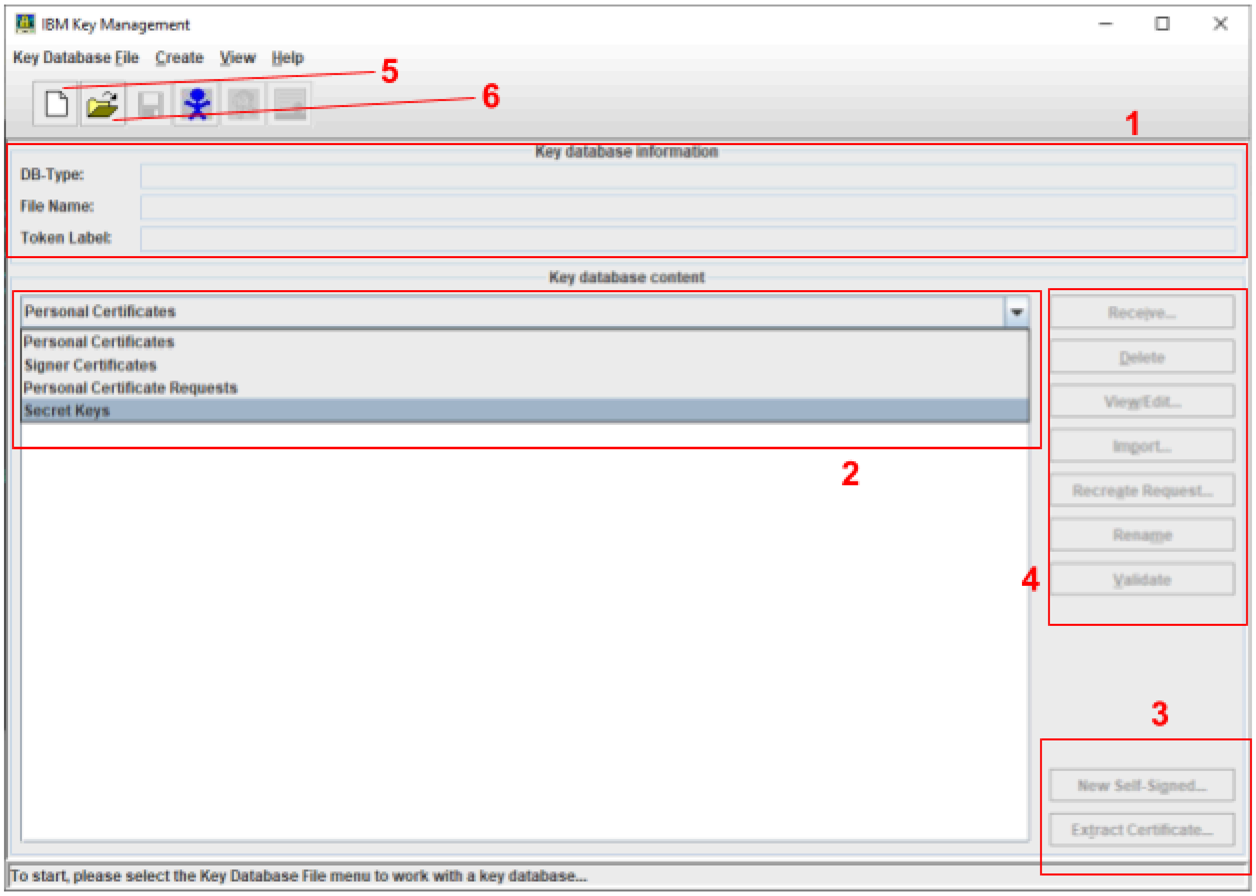
- **Java Keystore (JKS):** .jks - most common format for Java applications, also used by some other software.
- **PKCS12:** .pfx, .p12 - widely used format for storing certificates and private keys across various platforms.
- **PEM:** .pem - text-based format sometimes used for storing private keys, usually combined with other files for certificates.

Truststore:

- **Java Keystore (JKS):** .jks - can also be used for truststores in Java applications.
- **PEM:** .pem - common format for storing trusted CA certificates in text files.
- **JCE Truststore:** .jceks - specific format for truststores used with some Java Cryptographic Extension (JCE) libraries.

IBM Key Management GUI

1. Key Database Information
2. There are three views in iKeyman selectable from the drop-down arrow
   - Personal Certificates: This is where the Key Certificates will be shown
   - Signer Certificates: This is where the Intermediate and Root certificates, both yours and your trading partners', will be shown.
   - Personal Certificate Requests: This is where the Certificate Signing Requests are shown when created.
3. To create a Self Signed certificate / Extract a certificate(public) from selected key to file
4. Manage certificates by Receiving/Adding/Viewing/Deleting etc.
5. To Create a new KeyDatabase file that holds certificates/keys.
6. To Open a KeyDatabase file.



To start, please select the Key Database File menu to work with a key database...

Working with IBM KeyManagement Tool

We can manage / create Certificates by following actions

1. Creating a key pair and requesting a certificate from a Certificate Authority
2. Receiving a certificate into a key database
3. Creating a key pair and certificate request for self-signing
   Etc…

Implementing One-way authentication in App Connect

In a **one-way authentication**, the client needs to configure its trust store. This is because only the server sends its certificate to the client for verification. The client needs to have the server's certificate in its trust store to validate its authenticity and establish a secure connection.
Here's a breakdown of the roles:

- Server:
  - Generates and possesses its own self-signed certificate or a certificate issued by a trusted Certificate Authority (CA).
  - Sends its certificate to the client during the connection handshake.
- Client:
  - Verifies the server's certificate against its trust store.
  - If the certificate is valid and trusted, the connection is established securely.
  - Does not send its own certificate to the server in a one-way authentication.

If the server certificate isn't present or trusted within the client's trust store, the connection will fail due to failed verification.

Prerequisites :

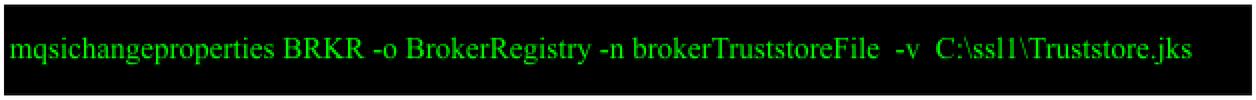
1. IBM Key Management Tool
2. App Connect Enterprise Toolkit
3. App Connect Enterprise Console

Scenario : Assuming app connect acts as Consumer to external service by requesting

In this case Integration team has to obtain public certificate(signed CA certificate) and add to it's trust store file from external service

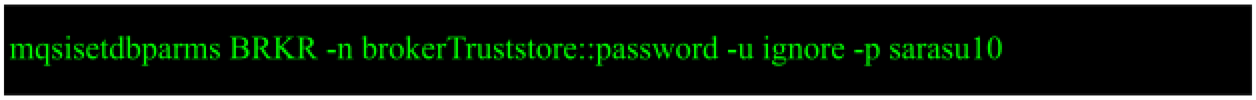
Steps :

1. Open IBM Key Management Tool by typing command on ACE Console
2. Open trust store key db file.
3. Select signed certificate from key database content drop down.
4. Add obtained public certificate by locating to its contained file.
5. Open ACE console add truststore file path to brokerTruststoreFile property on node/server.



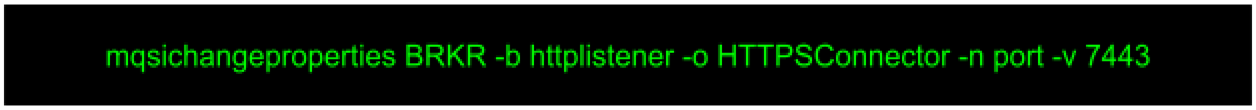
```
mqsichangeproperties BRKR -o BrokerRegistry -n brokerTruststoreFile  -v  C:\ssl1\Truststore.jks
```

6. Associate password for brokerTruststore file accessed by node/server



```
mqsisetdbparms BRKR -n brokerTruststore::password -u ignore -p sarasu10
```

7. Check httplistener port and specify if not provided by following



```
mqsichangeproperties BRKR -b httplistener -o HTTPSConnector -n port -v 7443
```

8. Go to browser and ensure by hitting url of your integration server

Frequently Asked Questions :

1. Is there no need to configure a trust store in case a server acts as a provider?
   Ans : yes.