Setting up Keystores

WSO2 products use asymmetric encryption by default for the authentication and protection of data. In asymmetric encryption, keystores (with private keys and public key certificates) and truststores (with only public key certificates) are created and stored for a product. It is possible to have multiple keystores so that the keys used for different use cases are kept unique. The following topics explain more details on keystores and truststores, and how they are used in WSO2 products.

- Understanding keystores and truststores
- Usage of keystores in WSO2 products
- Recommendations for setting up keystores in WSO2 products
- Managing keystores

Understanding keystores and truststores

A keystore is a repository (protected by a password) that holds the keys and certificates of a trust chain. There may be multiple trust chains (i.e., multiple keys with corresponding certificates) in one keystore. You use these artifacts for security purposes such as protecting sensitive information and establishing trust between your server and outside parties that connect to the server. The usage of keys and certificates contained in a keystore are explained below.

**Keys:** According to public-key cryptography, the concept of a key pair (public key and the corresponding private key) is used for protecting sensitive information and for authenticating the identity of external parties that communicate with your server. For example, the information that is encrypted in your server using the public key can only be decrypted using the corresponding private key. Therefore, if any party wants to decrypt this encrypted data, they should have the corresponding private key, which is usually kept as a secret (not publicly shared).

**Digital certificate:** When there is a key pair, it is also necessary to have a digital certificate to verify the identity of the keys. Typically, the public key of a key pair is embedded in this digital certificate, which also contains additional information such as the owner, validity, etc. of the keys. For example, if an external party wants to verify the integrity of data or validate the identity of the signer (by validating the digital signature), it is necessary for them to have this digital certificate of the signer.

**Trusted certificates and certificate signing authorities:** To establish trust, the digital certificate containing the public key should be signed by a trusted certificate signing authority (CA). You can generate self-signed certificates for the public key (thereby creating your own certifying authority), or you can get the certificates signed by the digital certificate of an external CA. When the certificate is signed by a reputed CA, all the parties that trust this CA will also trust the certificates signed by them. To establish maximum trust, it is important to have a root CA directly sign your public key certificate, or else, you can have an intermediate CA certificate (which is already signed by a root CA) sign your certificate. Therefore, in the later case, there can be a chain of CAs involved in signing your public key certificate. However, note that both types of public key certificates (self-signed or CA-signed) can be effectively used depending on the sensitivity of the information that is protected by the keys.

In summary, each trust chain entry in a keystore contains the following:

- A private key protected by a password.
- A digital certificate in which the public key (corresponding to the private key) is embedded.
- Additionally, If this public key certificate is not self-signed but signed by a Certificate Signing Authority (CA), an additional set of certificates (of the CAs involved in the signing process) will also be included. This may be just one additional certificate if the immediate CA certificate that was used to sign the public key certificate is of a Root CA. If the immediate certificate is not of a root CA, all the certificates of the intermediate CAs should also be included in the keystore.

The usage of a truststore in WSO2 products aligns with this concept of trust explained above. A truststore is just another repository that is protected by a password (similar to a keystore), which stores digital certificates. These certificates can be either of the following:

- Certificates of trusted third parties with which a software system intends to communicate directly.
- Certificates of reputed certificate signing authorities (CA) that can be used to validate the identity of untrusted
third parties that are being contacted. For example, consider a scenario where the exact certificate of the third party that the WSO2 server is attempting to contact is not in the truststore. In this scenario, if the third party has a CA-signed certificate and one of the certificates of its trust chain is already included in the WSO2 server's truststore, the trust is automatically granted and a successful SSL connection is established between the WSO2 server and the third party.

Default keystore and truststore in WSO2 products

All WSO2 products are by default shipped with a keystore file and truststore file (stored in the <PRODUCT_HOME>/repository/resources/security/ directory):

- **wso2carbon.jks**: This is the default keystore, which contains a private key and the self-signed public key certificate.
- **client-truststore.jks**: This is the default truststore, which contains certificates of reputed CAs that can validate the identity of third party systems. This truststore also contains the self-signed certificate of the default wso2carbon.jks keystore.

Usage of keystores in WSO2 products

In WSO2 products, asymmetric encryption is used by default for the following purposes:

- Authenticating the communication over Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocols.
- Encrypting sensitive data such as plain-text passwords found in both product-level and product feature-level configurations/configuration files using **secure vault**.
- Encrypting and signing SOAP messages using WS-Security.

The default keystore that is shipped with a WSO2 product (wso2carbon.jks) is by default configured for all of the above purposes. However, in a production environment, it is advised to set up several different keystores with separate trust chains for the above use cases.

Recommendations for setting up keystores in WSO2 products

Follow the recommendations given below when you set up your keystores.

- Maintain one primary keystore for encrypting sensitive internal data such as admin passwords and any other sensitive information found at both product-level and product feature-level configurations/configuration files.
- Maintain another secondary keystore, containing the server’s public key certificate for authenticating communication over SSL/TLS (for both Tomcat and Axis2 level HTTP connections).
- If your deployment contains multiple products, instances of the same product must use the same keystore for SSL. Different products can use different keystores for SSL, but it is not mandatory.
- It is recommended to use a CA-signed keystore for SSL communication; however, this is not mandatory. Even a self-signed certificate may suffice if it can be trusted by the clients.
- The keystore used for SSL must contain the same password for the Keystore and private key due to a Tomcat limitation.
- The primary keystore used for admin passwords and other data encryption requirements can be a self-signed one. There is no value added by using a CA-signed keystore for this purpose as it is not used for any external communication.
- The primary keystore’s public key certificate must have the **Data Encipherment** key usage to allow direct encipherment of raw data using its public key. This key usage is already included in the self-signed certificate that is included in the default wso2carbon.jks keystore. If the **Data Encipherment** key usage is not included in your public key certificate, the following error can occur when you attempt data encryption:
Exception in thread "main" org.wso2.ciphertool.CipherToolException:
Error initializing Cipher at
org.wso2.ciphertool.CipherTool.handleException(CipherTool.java:861) at
org.wso2.ciphertool.CipherTool.initCipher(CipherTool.java:202) at
org.wso2.ciphertool.CipherTool.main(CipherTool.java:80) Caused by:
java.security.InvalidKeyException: Wrong key usage at
javax.crypto.Cipher.init(DashoA13..) at
javax.crypto.Cipher.init(DashoA13..) at
org.wso2.ciphertool.CipherTool.initCipher(CipherTool.java:200) ... 1
more

• Optionally, you can set up separate keystores for message-level data encryption in WS-Security.

For information on creating new keystores with the required certificates, see Creating New Keystores, and for information on how to update configuration files in your product with keystore information, see Configuring Keystores in WSO2 Products.

Managing keystores

WSO2 products provide the facility to add keystores using the Management Console or using an XML configuration, and to import certificates to the keystore using the Management Console. The WSO2 keystore management feature provides a UI and an API to add and manage keystores. For example, when you apply WS-Security to Web services using the Management Console, you can select a keystore for encryption/signing processes from the uploaded keystores. The Management Console also allows you to view/delete keystores.

All the functions of keystore management are exposed via APIs. As a result, if you are writing a custom extension to a WSO2 product (e.g., for WSO2 ESB mediators), you can directly access the configured keystores using the API. The API hides the underlying complexity, allowing you to easily use it in third-party applications to manage their keystores as well.

This functionality is bundled with the following feature that is installed in your product.

**Name:** WSO2 Carbon - Security Management Feature

**Identifier:** org.wso2.carbon.security.mgt.feature.group