B

Managing Oracle Database Wallets and Certificates

You can use the `orapki` command line utility and sqlnet.ora parameters to manage public key infrastructure (PKI) elements.

- Introduction to Oracle Database Wallets and Certificates
  Oracle Database provides several types of public key infrastructure (PKI) elements (wallets and certificates), as well as tools to manage them.
- Managing Oracle Database Wallets and Certificates with the orapki Utility
  The `orapki` command-line utility is installed by default with the Oracle Database server.
- Managing Oracle Database Wallets
  The `orapki` command-line utility enables you to create and manage wallets before you add certificates to them.
- Managing Oracle Database Certificates
  After you create a wallet, you can associate certificates with it to validate the identities of entities that are associated with the wallet.
- Examples of Creating Wallets and Certificates Using orapki
  Examples of `orapki` commands include creating wallets, user certificates, and wallets with self-signed certificates, and exporting certificates.
- orapki Utility Commands Summary
  The `orapki` commands perform a variety of wallet, certificate revocation lists (CRL), and certificate management tasks.
- mkstore Utility Commands Summary
  The `mkstore` command line utility, available as part other Oracle Database client and server installations, enables you to create wallets and add credential secrets such as user names and passwords.

B.1 Introduction to Oracle Database Wallets and Certificates

Oracle Database provides several types of public key infrastructure (PKI) elements (wallets and certificates), as well as tools to manage them.

- About Oracle Database Wallets
  An Oracle Database wallet is a password-protected container that stores authentication and signing credentials, including private keys and certificates that enable database clients to communicate across an Oracle Database network.
- About Oracle Database Certificates
  An Oracle Database certificate (public key infrastructure (PKI) digital certificate) is a wallet component that validates the identity of an end entity in a public key or private key exchange that uses the wallet.
- About Certificate Authority (CA)
  A certificate authority (CA) is a trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are.

- Tools Used to Manage Oracle Database Wallets and Certificates
  Oracle Database provides different tools for managing wallets and certificates, depending on how the wallet will be used.
- General Process of Managing Oracle Database Wallets and Certificates
  Except for Transparent Data Encryption (TDE), you can use the `orapki` utility to create and manage Oracle Database wallets and certificates.
- Oracle Database Wallet Search Order
  The search order that Oracle Database uses to find wallets depends on the feature for which the wallet was created, such as Transparent Data Encryption (TDE).

B.1.1 About Oracle Database Wallets

An Oracle Database wallet is a password-protected container that stores authentication and signing credentials, including private keys and certificates that enable database clients to communicate across an Oracle Database network.

The authentication and signing credentials in a wallet are encrypted. Oracle Database clients can read and use wallets when the client connects to the database server. The database server can also read and use wallets when it connects with other services such as directory services. Before a wallet can be used, it must be "open", that is, made accessible by the database server that must read and use the wallet. Depending on how the wallet is created, the wallet must be either opened manually by a database administrator or it can be opened automatically.

Oracle Database provides the following use cases for wallet use:

- Outbound wallets, which are used by the database server to connect with outside services, such as Oracle wallets used for Oracle Database connections with Microsoft Active Directory and `UTL_HTTP`. These are created and managed with the `orapki` utility.
- Secure external password store (SEPS) wallets, which are used for clients only and are created only with the read/write permissions of the current user, so that other users cannot read this wallet.
- Transport Layer Security (TLS) wallets, for both server and clients. These are used for strong authentication.
- Transparent Data Encryption (TDE) wallets, which are used for servers and clients, and are called keystores. See *Oracle Database Advanced Security Guide*.

There are four types (or modes) of wallets: standard password-protected wallet (PKCS#12, which have the `.p12` file extension), and three types of auto-login wallets.

- **Password-protected wallets:** When you create this type of wallet, you must assign it a password. Later on, when you perform different tasks with this wallet, such as modifying it, you must provide the password. This type of wallet must be explicitly opened by a database administrator before it can be used. The password-protected wallet conforms to the PKCS#12 standard with a file name of `ewallet.p12`.
- **Single sign-on (SSO) auto-login wallets:** When you create an auto-login wallet, you must provide a password. An auto-login wallet allows encrypted storage of secrets such as passwords so they are not stored in clear text files. Oracle Database can read the secrets in the wallet without requiring a user to enter a password every time. This type is automatically opened by the database server that accesses it. An auto-login wallet is a read/write wallet that consists of both a PKCS #12 file called `ewallet.p12` and a single sign-on (SSO) file called `cwallet.sso`. Both files contain the same content except that the `ewallet.p12` is protected with a user password while `cwallet.sso` is protected with an obfuscated random password. When you use the Oracle wallet utilities (`orapki` and

`mkstore` (deprecated)) to modify auto-login wallets, you must provide the password that was used to create the `ewallet.p12` wallet file. (Any modification can happen only on the `ewallet.p12` file and the changes are internally applied to the corresponding `cwallet.sso` file. The `cwallet.sso` cannot be modified on its own.)
You can use auto-login wallets across different systems. If your environment does not require the extra security provided by a wallet that must be explicitly opened for use, then you can use an auto-login wallet. Auto-login wallets are ideal for unattended scenarios (for example, Oracle Data Guard standby databases).

- **Local single sign-on (LSSO) auto-login wallets:** This type is an auto-login wallet that is used only locally to the computer on which it was created. It cannot be opened on any computer other than the one on which it is created. It is a read/write wallet that does not require a user password. It is locked to the host name and user name that were in effect when it was created; it consists only of an SSO file called `cwallet.sso`.
  Local auto-login wallets are used for scenarios where additional security is required (that is, to limit the use of the auto-login for that computer) while supporting an unattended operation. You cannot use local auto-open wallets in Oracle Real Application Clusters (Oracle RAC)-enabled databases, because only shared wallets (in ACFS or ASM) are supported on those systems.
- **Auto-Login only (ALO or ESSO) wallet:** This wallet type is a read/write wallet that does not require a user password. It consists an SSO file called `cwallet.sso`.

All wallets that you create in this release of Oracle Database are in the PKCS#12 format. You can include the following security objects in a wallet:

- Certificates, which authenticate and validate user identities and encrypt data on communication channels. You can include the following types of certificates: trusted certificates, root certificates, user certificates, server certificates, private certificates, public certificates, and self-signed certificates.
- Certificates requests, which are requests submitted by an applicant to a CA to get an SSL certificate.
- Certificate revocation list (CRL), which is a list of digital certificates that have been revoked by the issuing certificate authority (CA).
- Secrets (such as passwords).
- For PKCS#11 wallets, specific PKCS#11 information, such as the path to the PKCS#11 library, tokens, smart cards, token passwords, and the certificate label on the token. The current standard is PKCS#12 and by default, the `orapki` utility creates wallets using this standard.
- For TDE keystores, a master encryption key, which is responsible for encrypting the data it is associated with, such as a table column, tablespace, or database. When you set the key for the wallet, you can specify an encryption algorithm for it, such as AES256. TDE keystores can also store secrets, such as user names and passwords.

> **Note:**
>
> ***Be careful about deleting wallets.*** Doing so can cause problems in the Oracle Database environment if the wallet is in use. If you want to delete a wallet, then back it up beforehand.

Related Topics

- Managing Oracle Database Wallets and Certificates with the orapki Utility
  The `orapki` command-line utility is installed by default with the Oracle Database server.
- Configuring Centrally Managed Users with Microsoft Active Directory
  Oracle Database can authenticate and authorize Microsoft Active Directory users with the database directly without intermediate directories or Oracle Enterprise User Security.
- Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
  Identity and Access Management (IAM) users can be configured to connect to an Oracle Database as a service (Oracle DBaaS) instance.
- Authenticating and Authorizing Microsoft Azure Users for Oracle Databases
  An Oracle database can be configured for Microsoft Azure users of Microsoft Entra ID (previously called Microsoft Azure AD) to connect using single sign-on authentication.
- Managing the Secure External Password Store for Password Credentials
  The secure external password store (SEPS) is a client-side wallet that is used to store password credentials.
- Configuring Transport Layer Security Encryption
  Use Transport Layer Security (TLS), a secure and standard protocol, to encrypt your database client and server connections.
- Deleting a Wallet
  You can delete wallets, but be cautious when doing so. Deleting a wallet that is in use can problems with the Oracle Database environment.

B.1.2 About Oracle Database Certificates

An Oracle Database certificate (public key infrastructure (PKI) digital certificate) is a wallet component that validates the identity of an end entity in a public key or private key exchange that uses the wallet.

The certificate is an International Telecommunications Union (ITU) x.509 v3 standard data structure that securely binds an identity to a public key. It is created when the public key of an entity is signed by a trusted identity, a certificate authority (CA). The certificate ensures that information in the entity is correct, and that the public key belongs to that entity. A certificate contains the name of the entity, identifying information, expiration date, and a public key. It is also likely to contain a serial number and information about the rights, uses, and privileges associated with the certificate. Finally, it contains information about the CA that issued it.

Oracle Database enables you to configure and work with the following types of certificates:

- **Certificate chain:** This is an ordered list of certificates that contain an end-user or subscriber certificate and its certificate authority certificates.
- **Trusted root certificate:** This type, which is mandatory, identifies the certificate authority (CA) that issued the server or user certificate. If the server presents its certificate to the client, then the client will not accept that certificate unless it has a trusted root certificate from the CA that issued the server certificate. The reverse is also true: the server only trusts the client certificate if the server has the trusted root certificate that issued the client certificate. The trusted root certificate is the top certificate in a certificate chain, which is an ordered list of certificate components that can comprise the following: server or user certificate, trusted certificate, public or private certificate. Because it is trusted, it enables you to keep customer information private and secure.
- **Private certificate:** This type identifies the private key on which the wallet was created. A private certificate is only used by the user or server and is never sent to any other users or servers. A trust certificate validates a signed private or public certificate.

- **Public certificate:** This type is identifies the public key on which the wallet is created, and is similar to private certificates. It is a digitally signed document that validates the name and authorization of a sender.
- **Server certificate:** This type, which is mandatory, identifies the database server that the wallet will use. It specifies which resources that a given server can have access to. It is sometimes used on devices that several servers share. Server certificates are typically issued to hosts or domains. There will always be a server certificate, even if that certificate is self-signed.
- **User certificate:** This type, which is optional, identifies the client that the wallet will use. It specifies which resources that a given user can have access to. It is sometimes used on devices that several users share. When different users log in, their profile and certificate are automatically loaded, granting them access to their required information. User certificates are used in the following cases:
  - For mutual Transport Layer Security (TLS), in which both ends of the communications channel must identify themselves
  - For PKI certificate authentication, in which the user certificate not only identifies the client, but also authenticates the server
- **Self-signed certificate:** This type is a public key certificate that is not issued by a CA. Configure self-signed certificates when there is no need for anyone to trust it, that is, you are only concerned with encryption. Even with a self-signed certificate, you still need the clients to connect. Therefore, the self-signed certificate is added to the client as a trusted certificate.

Following are some of the PKI elements that are related to certificates:

- **Certificate request:** The request has three parts: certification request information, a signature algorithm identifier, and a digital signature on the certification request information. The certification request information consists of the subject's distinguished name, public key, and an optional set of attributes. The attributes may provide additional information about the subject identity, such as postal address, or a challenge password by which the subject entity may later request certificate revocation. It is not mandatory to create a certificate request for the wallet. You can directly add a trusted certificate to the wallet or even a user certificate if a trusted certificate is already added.
- **Certificate revocation list (CRL):** This type is a signed data structure that contains a list of revoked certificates. The authenticity and integrity of the CRL is provided by a digital signature appended to it. Usually, the CRL signer is the same entity that signed the issued certificate. Typically, you create CRLs for user certificates. Because user certificates are held by users, it is not uncommon for them to be lost or stolen. When that happens, the issuing authority revokes them, and then publishes the revocation in the certificate revocation list that the services know not to trust the compromised certificates.

Related Topics

- About Certificate Authority (CA)
  A certificate authority (CA) is a trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are.

B.1.3 About Certificate Authority (CA)

A certificate authority (CA) is a trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are.

When it certifies a user, the CA first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The CA has its own certificate and public key which it

publishes. Servers and clients use these to verify signatures the certificate authority has made. A CA might be an external company that offers certificate services, or an internal organization such as a corporate management information systems (MIS) department. You must send the certificate request to this CA. The CA will send you a signed user certificate and its associated trusted certificate.

B.1.4 Tools Used to Manage Oracle Database Wallets and Certificates

Oracle Database provides different tools for managing wallets and certificates, depending on how the wallet will be used.

- `orapki` is a command-line Oracle utility that you can use to create wallets, and then add and manage certificates, certificate requests, and certificate revocation lists (CRLs) in the wallet.
- `mkstore` is a command-line Oracle utility that you can use to add secrets and credentials to the wallet and then manage them. It is available in the Oracle Database client. Starting in Oracle Database release 23ai, `mkstore` is deprecated. Oracle recommends that you use the `orapki` instead of `mkstore`.
- The `ADMINISTER KEY MANAGEMENT` statement provides a SQL*Plus interface for managing Transparent Data Encryption (TDE) keystores. TDE keystore management also provides data dictionary and dynamic views for finding information about keystores.
- Oracle Key Vault enables you to centrally manage existing keys and security objects within an enterprise.

> **Note:**
>
> Starting with Oracle Database 23ai, the Oracle Wallet Manager (OWM) is desupported.
> Oracle recommends using the `orapki` command line tool to replace OWM.

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Key Vault Administrator's Guide*

B.1.5 General Process of Managing Oracle Database Wallets and Certificates

Except for Transparent Data Encryption (TDE), you can use the `orapki` utility to create and manage Oracle Database wallets and certificates.

The general process is as follows:

1. Use the `orapki wallet create` command to create the wallet.
   For example, to create the wallet in the `$ORACLE_HOME/admin/`*`db_unique_name`*`/wallet` directory:

   ```
   orapki wallet create -wallet $ORACLE_HOME/admin/db_unique_name/wallet
   ```

2. Use the `orapki wallet add` command to generate a certificate request to associate with the wallet.
   For example, for a DN named `CN=server_dn,C=US`, using a key size of 2048 bits:

```
orapki wallet add -wallet $ORACLE_HOME/admin/db_unique_name/wallet
-dn 'CN=server_dn,C=US'
-keySize 2048
```

3. After the certificate request is generated, send it to the certificate authority (CA) that you want to use.
   You can export the certificate request to a file by using the `orapki wallet export` command, and share that file with CA to get a signed certificate.

   For example, to export a request called `creq.txt`:

```
orapki wallet export -wallet $ORACLE_HOME/admin/db_unique_name/wallet
-dn 'CN=server_dn,C=US'
-request $ORACLE_HOME/admin/db_unique_name/wallet/creq.txt
```

4. The CA generates your signed user certificate and its associated trusted certificate. At this stage, you are ready to start importing certificates into the wallet.
5. Use the `orapki wallet add` command to import all the trusted certificates into the wallet. If you do not add all the trusted certificates, then the `orapki add` command will fail.

   For example, to add a trusted certificate `trusted_cert.txt` to the wallet:

```
orapki wallet add -wallet $ORACLE_HOME/admin/db_unique_name/wallet
-trusted_cert -cert $ORACLE_HOME/wallet/trusted_cert.txt
```

6. Use the `orapki wallet add` command to import the user certificate into the wallet.
   For example, to import a user certificate that is in the `cert.txt` file:

```
orapki wallet add -wallet $ORACLE_HOME/admin/db_unique_name/wallet/
ewallet.p12
-user_cert
-cert $ORACLE_HOME/wallet/cert.txt
```

B.1.6 Oracle Database Wallet Search Order

The search order that Oracle Database uses to find wallets depends on the feature for which the wallet was created, such as Transparent Data Encryption (TDE).

The Oracle Database listener uses the following search path for the wallet, in this order:

1. `WALLET_LOCATION` parameter setting in connect string
2. `WALLET_LOCATION` parameter setting in the `sqlnet.ora` file
3. Wallet in the `$TNS_ADMIN` environment variable setting

The default wallet locations are as follows:

- Linux: `/etc/ORACLE/WALLETS/user_name`
- Windows: `C:\Users\user_name\\ORACLE\WALLETS`

See the following topics for information about various search orders for wallets:

- Centrally managed users (CMU) with Microsoft Active Directory: About Using a dsi.ora File
- Secure external password (SEP) wallets: TBA
- Transport Layer Security (TLS) server wallets: Oracle Wallet Search Order
- Transparent Data Encryption keystores: *Oracle Database Advanced Security Guide*
- Enterprise User Security wallets: *Oracle Database Enterprise User Security Administrator's Guide* (Note that Enterprise User Security is deprecated starting with Oracle Database 23ai.)

B.2 Managing Oracle Database Wallets and Certificates with the orapki Utility

The `orapki` command-line utility is installed by default with the Oracle Database server.

- About Managing Oracle Database Wallets and Certificates with the orapki Utility
  The `orapki` command-line utility enables you to create and manage wallets and certificates from the command line.
- orapki Utility Syntax
  The `orapki` utility syntax provides ways to create and manage wallets and certificates.

B.2.1 About Managing Oracle Database Wallets and Certificates with the orapki Utility

The `orapki` command-line utility enables you to create and manage wallets and certificates from the command line.

You can use `orapki` to perform the following tasks:

- Creating and viewing signed certificates for testing purposes
- Managing Oracle wallets (except for Transparent Data Encryption keystores):
  - Creating and displaying Oracle wallets
  - Adding and removing certificate requests
  - Adding and removing user certificates
  - Adding and removing trusted certificates
  - Importing and exporting the private key
  - Importing a PKCS12 file
  - Converting a JKS keystore to a PKCS12 file or vice versa
  - Exporting the certificates and certificate chain
- Managing certificate revocation lists (CRLs):
  - Renaming CRLs with a hash value for certificate validation
  - Uploading, listing, viewing, and deleting CRLs in Oracle Internet Directory

`orapki` enables you to automate these tasks by using scripts. Providing a way to incorporate the management of wallets, certificates, and certificate revocation lists (CRLs) into scripts makes it possible to automate many of the routine tasks of maintaining them.

You can use the `orapki` utility `wallet` module commands in scripts to automate the wallet creation process. For example, you can create password-protected wallets, auto-login wallets, auto-login-only wallets, or local auto-login wallets. You can create local auto-login wallets that are associated with PKCS#12 wallets that are local to the computer on which they were created and the user who created them. You can view wallets, import wallets, modify wallet passwords, and convert wallets to use the AES256 algorithm.

When you create a new wallet (any type), Oracle creates it as a version 6 wallet. If you modify an existing LSSO version 6 wallet, then orapki converts it to version 7. Starting in Oracle Database 23ai, version 6 of the local auto-login wallet is deprecated. You can check the version of the wallet by running the `orapki wallet display` command with the `ssvs` parameter, which displays the version of the wallet.

> **Note:**
>
> The `-wallet` parameter is mandatory for all `wallet` module commands.

Related Topics

- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

B.2.2 orapki Utility Syntax

The `orapki` utility syntax provides ways to create and manage wallets and certificates.

The syntax of the `orapki` command-line utility is as follows:

```
orapki module command -parameter value
```

In this specification, *`module`* can be `wallet` (Oracle wallet), `crl` (certificate revocation list), `cert` (PKI digital certificate), or `secretstore` (secrets and credentials). The available commands depend on the *`module`* you are using.

For example, if you are working with a `wallet`, then you can add a certificate or a key to the wallet with the `add` command. The following example adds the user certificate located at `/private/lhale/cert.txt` to the wallet located at `$ORACLE_HOME/admin/`*`db_unique_name`*`/wallet/ewallet.p12`:

```
orapki wallet add -wallet $ORACLE_HOME/admin/db_unique_name/wallet/ewallet.p12 -
user_cert -cert /private/lhale/cert.txt
```

B.3 Managing Oracle Database Wallets

The `orapki` command-line utility enables you to create and manage wallets before you add certificates to them.

- Creating a PKCS#12 Wallet
  You can use the `orapki` utility to create a PKCS#12 Oracle wallet.
- Importing a PKCS#12 Wallet
  You can use the `orapki` utility to import a PKCS#12 file into an existing wallet.

- Creating an Auto-Login-Only Wallet
  You can use the `orapki` utility to create an auto-login only wallet.
- Creating a Local Auto-Login Wallet
  The `orapki` utility can create a local auto-login wallet.
- Creating an Auto-Login Wallet That Is Associated with a PKCS#12 Wallet
  You can create an auto-login wallet that is associated with a PKCS#12 wallet.
- Viewing a Wallet
  You can use the `orapki` utility to view a wallet.
- Modifying the Password for a Wallet
  You can use the `orapki` utility to modify the password of a wallet.
- Converting an Oracle Wallet to Use the AES256 Algorithm
  By default, an Oracle wallet that was created with the `ADMINISTER KEY MANAGEMENT` or `ALTER SYSTEM` statement is encrypted with AES256.
- Deleting a Wallet
  You can delete wallets, but be cautious when doing so. Deleting a wallet that is in use can problems with the Oracle Database environment.

B.3.1 Creating a PKCS#12 Wallet

You can use the `orapki` utility to create a PKCS#12 Oracle wallet.

- To create an Oracle PKCS#12 wallet (`ewallet.p12`), use the `orapki wallet create` command.

```
orapki wallet create -wallet wallet_file_directory [-pwd password]
```

In this specification:

- `wallet` specifies the location in which to create the `ewallet.p12` wallet file.
- `pwd` is a new password to be assigned to the wallet. If you create an auto-login wallet later on, then it will require this password. If you do not provide a password using the `pwd` parameter, then you are prompted to enter and reenter the new password. For better security, enter the password at the prompt instead of entering it at the command line. When you create the password, follow these requirements:
  - Use no fewer than 8 characters. The maximum length is unlimited.
  - Use mixed alphanumeric characters.

B.3.2 Importing a PKCS#12 Wallet

You can use the `orapki` utility to import a PKCS#12 file into an existing wallet.

- To import an Oracle PKCS#12 wallet (`ewallet.p12`), use the `orapki wallet import_pkcs12` command.

```
orapki wallet import_pkcs12 -wallet wallet_file_directory [[-pwd password] | [-
auto_login_only]]
[-pkcs12file pkcs12_location] [-pkcs12pwd pkcs12_password]
```

In this specification:

- `pkcs12file` refers to the Oracle PKCS#12 wallet that to import into the *`wallet_file_directory`* location.
- `pkcs12Pwd` is the password of that wallet file.

B.3.3 Creating an Auto-Login-Only Wallet

You can use the `orapki` utility to create an auto-login only wallet.

- To create an auto-login only wallet (`cwallet.sso`), which does not need a password to open the wallet, use the `orapki wallet create` command.

  ```
  orapki wallet create -wallet wallet_file_directory -auto_login_only
  ```

Note the following:

- You can modify or delete the auto-login-only wallet without using a password. File system permissions provide the necessary security for such auto-login-only wallets.
- This command creates a `cwallet.sso` file.

B.3.4 Creating a Local Auto-Login Wallet

The `orapki` utility can create a local auto-login wallet.

Starting in Oracle Database 23ai, version 6 of the local auto-login wallet is deprecated, to be replaced with version 7.

- To create a local auto-login wallet that is local to both the computer on which it is created and the user who created it, use the `orapki wallet create` command.

  ```
  orapki wallet create -wallet wallet_file_directory -auto_login_local [-pwd
  wallet_password]
  ```

  In this specification, `pwd` is the password that was created when the PKCS#12 wallet was created. If no password is provided, then you are prompted to enter and reenter the new password. For better security, enter the password at the prompt instead of entering it at the command line.

This command does the following:

- Creates an auto-login wallet (`cwallet.sso`) file in the *wallet_file_directory*.
- Associates the auto-login wallet with a PKCS#12 wallet (`ewallet.p12`). If the `ewallet.p12` file does not exist, this command creates it.
- You cannot move local auto-login wallets to another computer. They must be used on the host on which they are created.
- Even though a local auto-login wallet does not need a password to open, you must supply the password for the associated PKCS#12 wallet in order to modify or delete the wallet. Any update to the PKCS#12 wallet also updates the associated auto-login wallet.

B.3.5 Creating an Auto-Login Wallet That Is Associated with a PKCS#12 Wallet

You can create an auto-login wallet that is associated with a PKCS#12 wallet.

- To create an auto-login wallet (`cwallet.sso`) that is associated with a PKCS#12 wallet (`ewallet.p12`), use the `orapki wallet create` command.

  ```
  orapki wallet create -wallet wallet_file_directory -auto_login [-pwd wallet_password]
  ```

  In this specification,

  - If the *wallet_file_directory* already contains a PKCS#12 wallet, then auto-login is enabled for it. You must supply the password for the existing PKCS#12 wallet in order to enable auto-login for it. If the *wallet_file_directory* does not contain a PKCS#12 wallet, then a new PKCS#12 wallet is created. You must create a password for the new PKCS#12 wallet. Follow these password creation requirements:
    * Use no fewer than 8 characters. The maximum length is unlimited.
    * Use mixed alphanumeric characters.
  - `pwd` is the PKCS#12 wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Note that the auto-login wallet does not need a password to open; it automatically uses the password of its associated PKCS#12 wallet. Therefore, you must supply the password for the associated PKCS#12 wallet to modify or delete the auto-login wallet. Any update to the PKCS#12 wallet also updates the associated auto-login wallet.

B.3.6 Viewing a Wallet

You can use the `orapki` utility to view a wallet.

This command displays the certificate requests, user certificates, trusted certificates, secret store entries, and credentials that are contained in the wallet.

- To view an Oracle wallet, use the `orapki wallet display` command.

  ```
  orapki wallet display -wallet wallet_file_directory
  ```

  Output similar to the following appears:

  ```
  Requested Certificates:
  User Certificates:
  Trusted Certificates:
  ```

B.3.7 Modifying the Password for a Wallet

You can use the `orapki` utility to modify the password of a wallet.

When you change the password of an auto-login wallet, and if that wallet is version 6, then Oracle Database automatically updates the wallet to version 7.

1. Use the `orapki wallet change_pwd` command to change the password.

   ```
   orapki wallet change_pwd -wallet wallet_file_directory [-oldpwd wallet_password ] [-
   newpwd wallet_password]
   ```

   This command changes the current wallet password to the new password. The command prompts you for the old and new passwords if no password is supplied at the command line. Change the password using the following requirements:

   - Use no fewer than 8 characters. The maximum length is unlimited.
   - Use mixed alphanumeric characters.

2. If this wallet uses an auto-login only wallet, then regenerate the auto-login only wallet.

   ```
   orapki wallet create -wallet wallet_file_directory -auto_login_only
   ```

B.3.8 Converting an Oracle Wallet to Use the AES256 Algorithm

By default, an Oracle wallet that was created with the `ADMINISTER KEY MANAGEMENT` or `ALTER SYSTEM` statement is encrypted with AES256.

If you are using an older wallet that is encrypted with 3DES instead of AES256, then you can use the `orapki convert` command to convert the wallet to use the AES256 algorithm, which is stronger than 3DES. Oracle wallets that are created with `orapki` are created with the AES256 algorithm by default.
Be aware that though the AES256 algorithm is stronger than 3DES, there will be some degradation in `orapki` operations if you use AES256.

- To change the wallet algorithm from 3DES to AES256, use the `orapki wallet convert` command.

  ```
  orapki wallet convert -wallet wallet_file_directory [-pwd wallet_password] -
  compat_v12
  ```

  In this specification:

  - `pwd` is the wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.
  - `compat_v12` performs the conversion from 3DES to AES256.

  You can check if the wallet has been converted from 3DES to AES356 by running the `openssl pkcs12` command. For example:

  ```
  openssl pkcs12 -in sample/ewallet.p12 -info
  Enter Import Password: password
  ```

  Output similar to the following appears. The `AES-256-CBC` value in the last line confirms that the wallet is encrypted with AES256.

  ```
  MAC: sha1, Iteration 10000
  MAC length: 20, salt length: 8
  PKCS7 Encrypted data: PBES2, PBKDF2, AES-256-CBC, Iteration 10000, PRF
  hmacWithSHA256
  ```

B.3.9 Deleting a Wallet

You can delete wallets, but be cautious when doing so. Deleting a wallet that is in use can problems with the Oracle Database environment.

1. Check the wallet contents to ensure that it is safe to delete it.

   It is important to check a wallet's contents because some wallets may have additional information that you were not aware of that is being used by the database. Use the following `orapki` command to check the contents of the wallet:

   ```
   orapki wallet display -wallet wallet_file_directory
   ```

2. Back up the wallet in case you may need it again.

   You should be able to easily recreate the wallet if it is needed again.

3. Delete the wallet.

The following example deletes a password-protected wallet:

```
orapki wallet delete -wallet $ORACLE_HOME/admin/db_unique_name/wallet
Enter password: wallet_password
```

To delete an auto-login wallet, include the `-sso` parameter:

```
orapki wallet delete -wallet $ORACLE_HOME/admin/db_unique_name/wallet -sso
Enter password: wallet_password
```

If you want to delete Transparent Data Encryption keystores, then see *Oracle Database Advanced Security Guide* for information about the dangers of deleting keystores.

B.4 Managing Oracle Database Certificates

After you create a wallet, you can associate certificates with it to validate the identities of entities that are associated with the wallet.

- Certificate Store Location for System Wallets
  System wallets are located in the certificate store location.
- Adding a Certificate Request to an Oracle Wallet
  You can use the `orapki` utility to add certificate requests to Oracle wallets.
- Creating Signed Certificates
  The `orapki` utility provides a way to sign user certificate requests by an intermediate or root key.
- Creating a Signed Certificate Using a Self-Signed Root
  This certificates creation method involves the use of an Oracle wallet with self signed certificate.
- Adding a Trusted Certificate to an Oracle Wallet
  You can use the `orapki` utility to add trusted certificates to an Oracle wallet.
- Adding a Root Certificate to an Oracle Wallet
  You can use the `orapki` utility to add a root certificate to an Oracle wallet.
- Adding Root Certificate Authority That Requires an Intermediate Certificate Using Microsoft Internet Explorer
  This procedure explains how to install a new or replacement root certificate authority (CA) by downloading it from Microsoft Explorer versions 5, 6, or 7.
- Adding a User Certificate to an Oracle Wallet
  You can use the `orapki` utility to add a user certificate to an Oracle wallet.
- Verifying Credentials on the Hardware Device That Uses a PKCS#11 Wallet
  You can verify credentials on the hardware device using the PKCS#11 wallet.
- Adding PKCS#11 Information to an Oracle Wallet
  A wallet that contains PKCS#11 information can be used like any Oracle wallet.
- Viewing a Certificate
  After you create a certificate, you can use the `orapki` utility to view it.
- Controlling MD5 and SHA-1 Certificate Use
  You can use the `sqlnet.ora` file to control whether MD5 and SHA-1 signed certificates are accepted.

- Certificate Import and Export Operations
  You can use `orapki` to import and export certificates.
- Management of Certificate Revocation Lists (CRLs) with orapki Utility
  You must manage certificate revocation lists (CRLs) with the `orapki` utility.

B.4.1 Certificate Store Location for System Wallets

System wallets are located in the certificate store location.

The default certificate store location depends on the platform. For Microsoft Windows, it is in the Microsoft Certificate Store for Microsoft Windows. For Linux, its locations are as follows:

- `/etc/pki/tls/cert.pem`
- `/etc/ssl/certs/ca-certificates.crt`
- `/etc/pki/tls/certs/ca-bundle.crt`
- `/etc/ssl/ca-bundle.pem`
- `/etc/pki/tls/cacert.pem`
- `/etc/pki/ca-trust/extracted/pem/tls-ca-bundle.pem`
- `/etc/ssl/cert.pem`

If the certificate authority (CA) is not in any of these locations, then you can create a symlink `/etc/pki/tls/cert.pem` pointing to the CA certificate file. Only PEM-formatted certificates are supported in all of the system certificate store locations.

B.4.2 Adding a Certificate Request to an Oracle Wallet

You can use the `orapki` utility to add certificate requests to Oracle wallets.

- To add a certificate request to an Oracle wallet, use the `orapki wallet add` command.

  ```
  orapki wallet add -wallet wallet_file_directory -dn user_dn -keySize 512|768|1024|
  2048|4096|8192|16384
  ```

  In this specification:

Table B-1 Parameter Descriptions of orapki wallet add

Parameter	Description
`wallet`	Specifies the location of the wallet to which you want to add a certificate request.
`dn`	Specifies the distinguished name of the certificate to add.

Table B-1 (Cont.) Parameter Descriptions of orapki wallet add

Parameter	Description
`keySize`	Specifies the key size in bits for the certificate. The size that you enter indicates the strength of security for the certificate. Values are as follows: – `512`: Included for backward compatibility and is supported in non-FIPS mode – `768`: Supported in non-FIPS mode – `1024`: Current default for non-FIPS certificate keys and is supported in non-FIPS mode – `2048`: Current default for FIPS certificate keys – `4096`: As needed per your site's requirements – `8192`: As needed per your site's requirements – `16384`: As needed per your site's requirements

To sign the request, export it with the `orapki wallet export` command.

Related Topics

- Exporting Certificates and Certificate Requests from an Oracle Wallet
  You can use the `orapki` utility to export certificates and certificate requests from an Oracle wallet.

B.4.3 Creating Signed Certificates

The `orapki` utility provides a way to sign user certificate requests by an intermediate or root key.

In most cases, this command is used to create a signed certificate for testing purposes, but it can be used for other reasons as well. It creates a signed certificate from the certificate request. A self-signed certificate is not issued or signed by a Certificate Authority (CA).

- To create a signed certificate, use the `orapki cert create` command.

```
orapki cert create [-wallet wallet_file_directory] -request
certificate_request_location -cert certificate_file -validity number_of_days [-pwd
wallet_password] [-cert_validation_mode strict|non-strict]
```

In this specification:

– `wallet` specifies the wallet containing the user certificate and private key that will be used to sign the certificate request.

– `validity` specifies the number of days, starting from the current date, that this certificate will be valid. Specifying a certificate and certificate request is mandatory for this command.

– `pwd` is the wallet password. If you omit this parameter, then you are prompted for the password. For better security, enter the password at this prompt.

– `cert_validation_mode` specifies if strict certificate validation, conforming to the RFC#5280 standard is (`strict`) or is not (`non-strict`) being used.

B.4.4 Creating a Signed Certificate Using a Self-Signed Root

This certificates creation method involves the use of an Oracle wallet with self signed certificate.

Using a certificate signed by a public Certificate Authority (CA) simplifies TLS connections because the root trust certificate for the database server is most likely already available in the default trust store on clients.

1. Create a wallet and add a self-signed root certificate to this wallet.
   a. Create the wallet as follows:

      Create the wallet in its own directory (for example, `wallet1`) under the wallet directory structure

      ```
      orapki wallet create -wallet wallet_file_directory/wallet1 -pwd
      wallet_password -auto_login
      ```

      The default algorithm is AES256.

   b. Add a self-signed certificate to this wallet.

      For example:

      ```
      orapki wallet add -wallet wallet_file_directory/wallet1
      -dn 'CN=sales.us.example.com, O=Oracle, L=Reading, ST=Texas,
      C=US' -self_signed -validity 3650 -keysize 2048 -sign_alg sha256
      -pwd wallet_password
      ```

2. Create a second wallet in its own directory (for example, `wallet2`) for the certificate.

   ```
   orapki wallet create -wallet wallet_file_directory/wallet2 -pwd
   wallet_password
   -auto_login
   ```

3. Add a certificate request to this second wallet and export it into a file.

   ```
   orapki wallet add -wallet wallet_file_directory/wallet2
   -dn 'CN=server_test,C=US' -keysize 2048 -pwd wallet_password

   orapki wallet export -wallet wallet_file_directory/wallet2
   -dn 'CN=server_test,C=US' -request creq.txt -pwd wallet_password
   ```

4. Use the first wallet with a self-signed root key to sign the certificate request `creq.txt`.

   The option `-sign_alg sha256` setting to specifies the SHA-2 algorithm. The file `usercert.txt` file will contain the SHA-2 certificate.

   ```
   orapki cert create -wallet wallet_file_directory/wallet1
   -request wallet_file_directory/wallet2/creq.txt -cert
   wallet_file_directory/wallet2/usercert.txt
   -sign_alg sha256 -validity 3650
   ```

5. Verify that the user certificate has been created with SHA-2 algorithm.

```
openssl x509 -in  wallet_file_directory/wallet2/usercert.txt -text
```

Output similar to the following appears:

```
Certificate:
Data:
Version: 1 (0x0)
Serial Number: 0 (0x0)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=US, ST=Texas, L=Reading, O=Oracle,
sales.us.example.com
Validity
Not Before: Aug 5 06:50:44 2023 GMT
Not After : Aug 2 06:50:44 2027 GMT
Subject: C=US, CN=server_test
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (2048 bit)
Modulus (2048 bit):
00:b0:36:ba:33:86:9f:f2:03:c0:13:b5:a2:99:09:

...

oU6jgrYfZkxcMMZMhnWKCpNBdA==
-----END CERTIFICATE-----
```

6. Export the self-signed certificate from the first wallet and import it as a trusted certificate into the second wallet.

To add the signed certificate into the original (second wallet), first you must import the root trust certificate and any intermediate trust certificates in hierarchical order before you can add the newly signed user certificate. This example uses the root private key to sign the user certificate, so you just need to export the self-signed root certificate from the first wallet and then import it as a trusted certificate into the second wallet.

```
orapki wallet export -wallet wallet_file_directory/wallet1
-dn 'CN=sales.us.example.com, O=Oracle, L=Reading, ST=Texas, C=US'
-cert self_cert.crt

orapki wallet add -wallet wallet_file_directory/wallet2 -trusted_cert -
cert
/wallet_file_directory/wallet1/self_cert.crt -pwd wallet_password
```

7. Import the certificate file `usercert.txt` into the second wallet.

```
orapki wallet add -wallet wallet_file_directory/wallet2 -user_cert
-cert wallet_file_directory/wallet2/usercert.txt
-sign_alg sha256 -pwd wallet_password
```

8. In the domain for the wallet and certificate, display the wallet to confirm.

```
[sales]wallet_file_directory/wallet2> orapki
wallet display -wallet .
```

Output similar to the following should appear:

```
Requested Certificates:
User Certificates:
Subject: CN=server_test,C=US
Trusted Certificates:
Subject: O=Oracle\, Inc.,C=US,
Inc.,C=US
Subject: CN=GTE CyberTrust Global Root,
Inc.,O=GTE Corporation,C=US
```

B.4.5 Adding a Trusted Certificate to an Oracle Wallet

You can use the `orapki` utility to add trusted certificates to an Oracle wallet.

This command adds a trusted certificate to the specified location (`-cert` *`certificate_file_directory`*), to a wallet. You must add all trusted certificates in the certificate chain of a user certificate before adding a user certificate, or the command to add the user certificate will fail.

- To add a trusted certificate to an Oracle wallet, use the `orapki wallet add` command.

  ```
  orapki wallet add -wallet wallet_file_directory -trusted_cert -cert certificate_file
  [-pwd wallet_password]
  ```

  If you omit the `-pwd` parameter, then you are prompted to enter the wallet password. For better security, enter the password at this prompt.

B.4.6 Adding a Root Certificate to an Oracle Wallet

You can use the `orapki` utility to add a root certificate to an Oracle wallet.

This command creates a new self-signed (root) certificate and adds it to the wallet.

- To add a root certificate to an Oracle wallet, use the `orapki wallet add` command.

  ```
  orapki wallet add -wallet wallet_file_directory -dn certificate_dn -keySize 512|768|
  1024|2048|4096|8192|16384 -self_signed -validity number_of_days
   [-pwd wallet_password] [-cert_validation_mode strict|non-strict]
  ```

  In this specification:

  - `validity` specifies the number of days, starting from the current date, that this certificate will be valid. This parameter is mandatory.
  - `keySize` specifies the key size in bits of the requested certificate. The size that you enter indicates the strength of security for the certificate. Values are as follows:
    - `512`: Included for backward compatibility and is supported in non-FIPS mode
    - `768`: Supported in non-FIPS mode
    - `1024`: Current default for non-FIPS certificate keys and is supported in non-FIPS mode
    - `2048`: Current default for FIPS certificate keys
    - `4096`: As needed per your site's requirements
    - `8192`: As needed per your site's requirements

    - `16384`: As needed per your site's requirements
  - `pwd` is the wallet password. If you omit this parameter, then you are prompted for the password. For better security, enter the password at this prompt.
  - `cert_validation_mode` specifies if strict certificate validation, conforming to the RFC#5280 standard is (`strict`) or is not (`non-strict`) being used.

B.4.7 Adding Root Certificate Authority That Requires an Intermediate Certificate Using Microsoft Internet Explorer

This procedure explains how to install a new or replacement root certificate authority (CA) by downloading it from Microsoft Explorer versions 5, 6, or 7.

1. In Internet Explorer, select **Tools**, then **Internet Options**, then **Content**, then **Certificates**.
2. Selct the **Trusted Root Certification Authorities** tab.
3. Select **Issued to: …**.
4. Click **Export**.
5. In the wizard that opens, select **Next**, then **Select Base-64 encoded X.509 (.CER)**.
6. Enter a file name and select **Finish**.

B.4.8 Adding a User Certificate to an Oracle Wallet

You can use the `orapki` utility to add a user certificate to an Oracle wallet.

1. Ensure that you have added to the wallet all the trust certificates that make up the certificate chain.

   If all trusted certificates are not installed in the wallet before you add the user certificate, then adding the user certificate will fail.
2. Use the `orapki wallet add` command to add the user certificate to the wallet.

   ```
   orapki wallet add -wallet wallet_file_directory -user_cert -cert
   certificate_file_directory [-pwd wallet_password]
   ```

   If you omit the `-pwd` parameter, then you are prompted to enter the wallet password. For better security, enter the password at this prompt.

B.4.9 Verifying Credentials on the Hardware Device That Uses a PKCS#11 Wallet

You can verify credentials on the hardware device using the PKCS#11 wallet.

- To verify the credential details, use the `orapki wallet p11_verify` command.

  ```
  orapki wallet p11_verify -wallet wallet_file_directory [-pwd wallet_password]
  ```

  `pwd` is the wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line

B.4.10 Adding PKCS#11 Information to an Oracle Wallet

A wallet that contains PKCS#11 information can be used like any Oracle wallet.

The private keys for this type of wallet are stored on a hardware device. Hardware devices maintain the private key and perform cryptographic operations using the private key. Therefore, the private key is never needed outside of the hardware device boundary.

- To add PKCS#11 information to a wallet, use the `orapki wallet p11_add` command.

  ```
  orapki wallet p11_add -wallet wallet_file_directory -p11_lib pkcs11Lib
  [-p11_tokenlabel tokenLabel] [-p11_tokenpw tokenPassphrase]
  [-p11_certlabel certLabel] [-pwd wallet_password]
  ```

  In this specification:

  - `p11_lib` specifies the path to the PKCS#11 library. This includes the library file name.
  - `p11_tokenlabel` specifies the token or smart card used on the device. Use this when there are multiple tokens on the device. Token labels are set using vendor tools.
  - `p11_tokenpw` specifies the password that is used to access the token. Token passwords are set using vendor tools.
  - `p11_certlabel` is used to specify a certificate label on the token. Use this when a token contains multiple certificates. Certificate labels are set using vendor tools.
  - `pwd` is the wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

B.4.11 Viewing a Certificate

After you create a certificate, you can use the `orapki` utility to view it.

- To view a certificate, use the `orapki cert display` command.

  ```
  orapki cert display -cert certificate_file_directory [-complete]
  ```

  In this specification:

  - `summary` displays the certificate and its expiration date.
  - `complete` displays additional certificate information, including the serial number and public key.

B.4.12 Controlling MD5 and SHA-1 Certificate Use

You can use the `sqlnet.ora` file to control whether MD5 and SHA-1 signed certificates are accepted.

To control whether the MD5 and SHA-1 signed certificates are accepted, you can edit the `sqlnet.ora` file to enable or disable their use.

> **Note:**
>
> MD5 is deprecated in this release.

1. Log in to the server where the Oracle database resides.
2. Edit the `sqlnet.ora` file.

   By default, the `sqlnet.ora` file is located in the `$ORACLE_HOME/dbs` directory or in the location set by the `TNS_ADMIN` environment variable.
3. Set the following parameters:
   - `ACCEPT_MD5_CERTS` controls the use of MD5 certificates. The default is `FALSE`. This parameter replaces the `ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES` environment variable.
   - `ACCEPT_SHA1_CERTS` controls the use of SHA-1 certificates. The default is `TRUE`.

B.4.13 Certificate Import and Export Operations

You can use `orapki` to import and export certificates.

- Importing a User-Supplied or Trusted Certificate into an Oracle Wallet
  You can add a user-supplied or trusted certificate to an Oracle wallet.
- Exporting Certificates and Certificate Requests from an Oracle Wallet
  You can use the `orapki` utility to export certificates and certificate requests from an Oracle wallet.

B.4.13.1 Importing a User-Supplied or Trusted Certificate into an Oracle Wallet

You can add a user-supplied or trusted certificate to an Oracle wallet.

- Use the `orapki wallet add -wallet` command as follows:
  - To add a trusted certificate to an Oracle wallet, use the `-trusted_cert` parameter.

    ```
    orapki wallet add -wallet wallet_file_directory [-pwd wallet_password] -
    trusted_cert -cert root_and/or_intermediate_certificate_file
    ```
  - To add a user-created certificate to an Oracle wallet, use the `-user_cert` parameter.

    ```
    orapki wallet add -wallet wallet_file_directory [-pwd wallet_password] -
    user_cert -cert user_certificate_file
    ```

  In this specification, `pwd` is the wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

B.4.13.2 Exporting Certificates and Certificate Requests from an Oracle Wallet

You can use the `orapki` utility to export certificates and certificate requests from an Oracle wallet.

- Depending on the type of certificate that you want to export from a wallet, use the `orapki wallet export` command.
  - To export a certificate with the subject's distinguished name (`-dn`) to a file that is specified by the `-cert` parameter:

    ```
    orapki wallet export -wallet wallet_file_directory -dn certificate_dn -cert
    certificate_filename
    ```

`dn` specifies the distinguished name of the certificate. In the case of a multi-valued DN, the order in which the individual DN values are stored in the wallet is uncertain. To find the correct DN that you want, run `orapki wallet display`.

– To export a certificate with an alias:

```
orapki wallet export -wallet wallet_file_directory -alias alias_name -cert
certificate_filename
```

– To export a certificate request with the subject's distinguished name (`-dn`) to a file that is specified by the `-request` parameter:

```
orapki wallet export -wallet wallet_file_directory -dn certificate_request_dn -
request certificate_request_filename
```

– To export private keys, use the following syntax:

```
orapki export_private_key -wallet wallet_file_directory -pvtkeyfile pvt_key_file
-alias pvt_key_alias -pvtkeypwd pvt_key_password
```

Related Topics

- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

B.4.14 Management of Certificate Revocation Lists (CRLs) with orapki Utility

You must manage certificate revocation lists (CRLs) with the `orapki` utility.

This utility creates a hashed value of the CRL issuer's name to identify the CRLs location in your system. If you do not use `orapki`, your Oracle server cannot locate CRLs to validate PKI digital certificates.

Related Topics

- Certificate Revocation List Management
  Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.

B.5 Examples of Creating Wallets and Certificates Using orapki

Examples of `orapki` commands include creating wallets, user certificates, and wallets with self-signed certificates, and exporting certificates.

- Example: Wallet with a Self-Signed Certificate and Export of the Certificate
  The `orapki wallet add` command can create a wallet with a self-signed certificate; the `orapki wallet export` can export the certificate.
- Example: Creating a Wallet and a User Certificate
  The `orapki` utility can create wallets and user certificates.

B.5.1 Example: Wallet with a Self-Signed Certificate and Export of the Certificate

The `orapki wallet add` command can create a wallet with a self-signed certificate; the `orapki wallet export` can export the certificate.

The following example illustrates the steps to create a wallet with a self-signed certificate, view the wallet, and then export the certificate to a file.

Example B-1 Creating a Wallet with a Self-Signed Certificate and Exporting the Certificate

1. Create a wallet.

   For example:

   ```
   orapki wallet create -wallet /private/user/orapki_use/root
   Enter password: new_password
   Enter password again: new_password
   ```

   The wallet is created at the location, `/private/user/orapki_use/root`.

2. Add a self-signed certificate to the wallet.

   ```
   orapki wallet add -wallet /private/user/orapki_use/root -dn
   'CN=root_test,C=US' -keysize 2048 -self_signed -validity 3650
   ```

   This creates a self-signed certificate with a validity of 3650 days. The distinguished name of the subject is `CN=root_test,C=US`. The key size for the certificate is 2048 bits.

3. View the wallet to check that the certificate is contained in the wallet.

   ```
   orapki wallet display -wallet /private/user/orapki_use/root
   ```

4. Export the certificate.

   ```
   orapki wallet export -wallet /private/user/orapki_use/root -dn
   'CN=root_test,C=US' -cert /private/user/orapki_use/root/b64certificate.txt
   ```

   This exports the self-signed certificate to the file, `b64certificate.txt`. Note that the distinguished name used is the same as in step 2.

B.5.2 Example: Creating a Wallet and a User Certificate

The `orapki` utility can create wallets and user certificates.

The following steps illustrate creating a wallet, creating a certificate request, exporting the certificate request, creating a signed certificate from the request for testing, viewing the certificate, adding a trusted certificate to the wallet and adding a user certificate to the wallet.

Example B-2 Creating a Wallet and a User Certificate

1. Create a wallet with auto-login enabled.

   For example:

   ```
   orapki wallet create -wallet /private/user/orapki_use/server -auto_login
   Enter wallet password: password
   ```

2. Add a certificate request to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -dn
'CN=server_test,C=US' -keysize 2048
```

This command adds a certificate request to the wallet that was created (`ewallet.p12`). The distinguished name of the subject is `CN=server_test,C=US`. The key size specified is 2048 bits, which sets it to a secure level.

3. Export the certificate request to a file.

```
orapki wallet export -wallet /private/user/orapki_use/server -dn
'CN=server_test,C=US' -request /private/user/orapki_use/server/creq.txt
```

This command exports the certificate request to the specified file, which is `creq.txt` in this case.

4. Create a signed certificate from the request for test purposes.

```
orapki cert create -wallet /private/user/orapki_use/root -request /private/user/
orapki_use/server/creq.txt -cert /private/user/orapki_use/server/cert.txt -validity
3650
```

This command creates a certificate, `cert.txt` with a validity of 3650 days. The certificate is created from the certificate request generated in the preceding step.

5. View the certificate.

```
orapki cert display -cert /private/user/orapki_use/server/cert.txt -complete
```

This command displays the certificate generated in the preceding step. The `-complete` option enables you to display additional certificate information, including the serial number and public key.

6. Add a trusted certificate to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -trusted_cert -
cert /private/user/orapki_use/root/b64certificate.txt
```

This command adds a trusted certificate, `b64certificate.txt` to the `ewallet.p12` wallet. You must add all trusted certificates in the certificate chain of a user certificate before adding a user certificate.

7. Add a user certificate to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -user_cert -
cert /private/user/orapki_use/server/cert.txt
```

This command adds the user certificate, `cert.txt` to the `ewallet.p12` wallet.

B.6 orapki Utility Commands Summary

The `orapki` commands perform a variety of wallet, certificate revocation lists (CRL), and certificate management tasks.

- orapki cert create
  The `orapki cert create` command creates a signed certificate for testing purposes.
- orapki cert display
  The `orapki cert display` command displays details of a specified certificate.
- orapki crl delete
  The `orapki crl delete` command deletes a certificate revocation list (CRL) that is stored in Oracle Internet Directory.

- orapki crl display
  The `orapki crl display` command displays a specified certificate revocation list (CRL) that is stored in Oracle Internet Directory.
- orapki crl hash
  The `orapki crl hash` command generates a hash value of the certificate revocation list (CRL) issuer to identify the CRL file system location for certificate validation.
- orapki crl list
  The `orapki crl list` command displays a list of certificate revocation lists (CRLs) that are stored in Oracle Internet Directory.
- orapki crl upload
  The `orapki crl upload` command uploads a certificate revocation list (CRL) to the CRL subtree in Oracle Internet Directory.
- orapki secretstore create_credential
  The `orapki secretstore create_credential` command creates database connection credentials in the wallet.
- orapki secretstore create_entry
  The `orapki secretstore create_entry` command stores a secret entries against an alias in a wallet.
- orapki secretstore create_user_credential
  The `orapki secretstore create_user_credential` command creates a credential object that is referenced by an alias that is constituted from a map and key name.
- orapki secretstore delete_credential
  The `orapki secretstore delete_credential` command deletes database connection credentials from a wallet.
- orapki secretstore delete_entry
  The `orapki secretstore delete_entry` command deletes the secret entries for an alias from a wallet.
- orapki secretstore delete_user_credential
  The `orapki secretstore delete_user_credential` command deletes the credential object that is referenced by the alias that was constituted from the map and key name.
- orapki secretstore list_credentials
  The `orapki secretstore list_credentials` command lists the contents of the external password store.
- orapki secretstore list_entries
  The `orapki secretstore list_entries` command lists the identifiers in a wallet.
- orapki secretstore list_entries_unsorted
  The `orapki secretstore list_entries_unsorted` command lists the identifiers in a wallet in unsorted order.
- orapki secretstore modify_credential
  The `orapki secretstore modify_credential` command modifies database connection credentials in the wallet.
- orapki secretstore modify_entry
  The `orapki secretstore modify_entry` command modifies the secret entry for an alias in a wallet.
- orapki secretstore modify_user_credential
  The `orapki secretstore modify_user_credential` command modifies a credential object that is referenced by an alias that was constituted from a map and key name.

- orapki secretstore view_entry
  The `orapki secretstore view_entry` command lists the secret entries for an alias in a wallet.
- orapki wallet add
  The `orapki wallet add` command adds certificate requests and certificates to an Oracle wallet.
- orapki wallet change_pwd
  The `orapki wallet change_pwd` command changes the password for a wallet.
- orapki wallet convert
  The `orapki wallet convert` command converts the 3DES algorithm in an Oracle wallet to use the AES256 algorithm.
- orapki wallet create
  The `orapki wallet create` command creates an Oracle wallet or enables auto-login for an Oracle wallet.
- orapki wallet delete
  The `orapki wallet delete` command deletes an Oracle wallet.
- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.
- orapki wallet export
  The `orapki wallet export` command exports certificate requests and certificates from an Oracle wallet.
- orapki wallet export_private_key
  The `orapki wallet export_private_key` command exports a private key from a wallet.
- orapki wallet import_pkcs12
  The `orapki wallet import_pkcs12` command imports a PKCS #12 file into the wallet. Only the latest valid certificate for each unique private key in a PKCS#12 file will be imported into an Oracle wallet. If a private key already exists in the wallet, its associated certificate chain will be skipped.
- orapki wallet import_private_key
  The `orapki wallet import_private_key` command imports a private key into a wallet.
- orapki wallet jks_to_pkcs12
  The `orapki wallet jks_to_pkcs12` command converts a Java keystore to PKCS #12 format for the storage of certificate information.
- orapki wallet pkcs12_to_jks
  The `orapki wallet pkcs12_to_jks` command converts a PKCS #12 keystore to a Java keystore for the storage of certificate information.
- orapki wallet remove
  The `orapki wallet remove` command removes certificates and certificate requests from the wallet.

B.6.1 orapki cert create

The `orapki cert create` command creates a signed certificate for testing purposes.

Syntax

```
orapki cert create [-wallet wallet_file_directory] -request certificate_request_location
-cert certificate_file_directory -validity number_of_days [-cert_validation_mode strict|
non-strict]
```

- `wallet` specifies the location of the wallet that contains the user certificate and private key that will be used to sign the certificate request.
- `request` specifies the location of the certificate request for the certificate you are creating.
- `cert` specifies the directory location where the tool places the new signed certificate.
- `validity` specifies the number of days, starting from the current date, that this certificate will be valid.
- `cert_validation_mode` specifies if strict certificate validation, conforming to the RFC#5280 standard is (`strict`) or is not (`non-strict`) being used.

Example

```
orapki cert create -wallet $ORACLE_HOME/admin/db_unique_name/wallet
-request $ORACLE_HOME/admin/db_unique_name/wallet/cert_reqs
-cert $ORACLE_HOME/admin/db_unique_name/wallet/certs
-validity 365 -summary
-cert_validation_mode strict
```

B.6.2 orapki cert display

The `orapki cert display` command displays details of a specified certificate.

Syntax

```
orapki cert display -cert certificate_file_directory [-complete]
```

- `cert` specifies the location of the certificate you want to display.
- `summary|complete` display the following information:
  - `summary` displays the certificate and its expiration date.
  - `complete` displays additional certificate information, including the serial number and public key.

Example

```
orapki cert display -wallet $ORACLE_HOME/admin/db_unique_name/wallet/certs
```

B.6.3 orapki crl delete

The `orapki crl delete` command deletes a certificate revocation list (CRL) that is stored in Oracle Internet Directory.

The user who deletes the CRLs from the directory by using `orapki` must be a member of the `CRLAdmins` (`cn=CRLAdmins,cn=groups,%s_OracleContextDN%`) directory group.

Syntax

```
orapki crl delete -issuer issuer_name -ldap hostname:ssl_port -user user_name [-wallet
wallet_file_directory] [-summary]
```

- `issuer` specifies the name of the certificate authority (CA) who issued the CRL.
- `ldap` specifies the host name and SSL port for the directory where the CRLs are to be deleted. Note that this must be a directory SSL port (uploaded to Oracle Internet Directory) with no authentication.
- `user` specifies the user name of the directory user who has permission to delete CRLs from the CRL subtree in the directory.
- `wallet` specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to deleting it from the directory.
- `summary` displays the CRL LDAP entry that was deleted.

Example

```
orapki crl delete -issuer psmith
-ldap hr_db:4415
-user psmith
-wallet $ORACLE_HOME/admin/db_unique_name/wallet
-summary
```

Related Topics

- Uploading CRLs to Oracle Internet Directory
  Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.

B.6.4 orapki crl display

The `orapki crl display` command displays a specified certificate revocation list (CRL) that is stored in Oracle Internet Directory.

Syntax

```
orapki crl display -crl crl_location [-wallet wallet_file_directory] [-summary|-complete]
```

- `crl` parameter specifies the location of the CRL in the directory. It is convenient to paste the CRL location from the list that displays when you use the `orapki crl list` command.
- `wallet` (optional) specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to displaying it.

- `summary` and `complete` display the following information:
  - `summary` provides a listing that contains the CRL issuer's name and the validity period of the CRL.
  - `complete` provides a list of all revoked certificates that the CRL contains. The output for this option may take a long time to display, depending on the size of the CRL.

Example

```
orapki  crl display -crl $ORACLE_HOME/admin/db_unique_name/wallet/crls
-wallet $ORACLE_HOME/admin/db_unique_name/wallet
-summary
```

Related Topics

- orapki crl list
  The `orapki crl list` command displays a list of certificate revocation lists (CRLs) that are stored in Oracle Internet Directory.

B.6.5 orapki crl hash

The `orapki crl hash` command generates a hash value of the certificate revocation list (CRL) issuer to identify the CRL file system location for certificate validation.

Syntax

```
orapki crl hash -crl crl_filename|URL [-wallet wallet_file_directory] [-symlink|-copy]
crl_directory [-summary]
```

- `crl` specifies the file name that contains the CRL or the URL where it can be found.
- `wallet` (optional) specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to uploading it to the directory.
- Depending on the operating system, use either the `-symlink` or the `-copy` parameter:
  - (UNIX) `symlink` creates a symbolic link to the CRL at the `crl_directory` location
  - (Windows) `copy` creates a copy of the CRL at the `crl_directory` location
- `summary` displays the CRL issuer's name.

Example

```
orapki crl hash -crl db_cert_rev
-wallet $ORACLE_HOME/admin/db_unique_name/wallet
-copy
-$ORACLE_HOME/admin/db_unique_name/wallet/crls
-summary
```

B.6.6 orapki crl list

The `orapki crl list` command displays a list of certificate revocation lists (CRLs) that are stored in Oracle Internet Directory.

Syntax

This command is useful for browsing to locate a particular CRL to view or download to your local file system.

```
orapki crl list -ldap hostname:ssl_port
```

`ldap` specifies the host name and SSL port for the directory server from where you want to list CRLs. Note that this must be a directory SSL port with no authentication.

Example

```
orapki crl list -ldap hr_db:4415
```

Related Topics

- Uploading CRLs to Oracle Internet Directory
  Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.

B.6.7 orapki crl upload

The `orapki crl upload` command uploads a certificate revocation list (CRL) to the CRL subtree in Oracle Internet Directory.

Note that you must be a member of the directory administrative group `CRLAdmins` (`cn=CRLAdmins,cn=groups,%s_OracleContextDN%`) to upload CRLs to the directory.

Syntax

```
orapki crl upload -crl crl_location -ldap hostname:ssl_port -user username [-wallet
wallet_file_directory] [-summary]
```

- `crl` specifies the directory location or the URL where the CRL is located that you are uploading to the directory.
- `ldap` specifies the host name and SSL port for the directory where you are uploading the CRLs. Note that this must be a directory SSL port with no authentication.
- `user` specifies the user name of the directory user who has permission to add CRLs to the CRL subtree in the directory.
- `wallet` specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. This is an optional parameter. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to uploading it to the directory.
- `summary` displays the CRL issuer's name and the LDAP entry where the CRL is stored in the directory.

Example

```
orapki  crl upload -crl $ORACLE_HOME/admin/db_unique_name/wallet/crls
-ldap hr_db:4415
-user psmith
-wallet $ORACLE_HOME/admin/db_unique_name/wallet
```

Related Topics

- Uploading CRLs to Oracle Internet Directory
  Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.

B.6.8 orapki secretstore create_credential

The `orapki secretstore create_credential` command creates database connection credentials in the wallet.

Syntax

```
orapki secretstore create_credential [-wallet wallet_file_directory] [-pwd
wallet_password]
[-default | -connect_string db_connect_string]
[-username user_name] [-password user_password]
```

- `wallet` specifies the path to the wallet directory where you want to store the credential. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `connect_string` can be the TNS alias that you use to specify the database in the `tnsnames.ora` file or any service name you use to identify the database on an Oracle Database network.
- `default` can be used instead of `connect_string` to add default credentials if the `connect_string` is neither available nor required. It is used to more conveniently set the default `username` and `password`.
- `username` and `password` are the database login credentials. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example B-3 Create credentials with a connect string

```
orapki secretstore create_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-connect_string sales.us.example.com -username pfitch
Enter wallet password: wallet_password
Enter user password: user_password
```

Example B-4 Create default credentials

```
orapki secretstore create_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-default -username pfitch -password sample_pass1
```

B.6.9 orapki secretstore create_entry

The `orapki secretstore create_entry` command stores a secret entries against an alias in a wallet.

Syntax

```
orapki secretstore create_entry [-wallet wallet_file_directory] [-pwd wallet_password]
[-alias alias] [-secret secret]
```

- `wallet` specifies the location of the wallet that will contain the secret entries for the specified alias. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `alias` specifies the name of the alias in which you want to store the secret entries.
- `secret` specifies the secret text that you want to store.

Example

```
orapki secretstore create_entry -wallet $ORACLE_HOME/admin/db_unique_name/
wallet
-alias db_alias -secret Time2Laugh@
Enter wallet password: wallet_password
```

B.6.10 orapki secretstore create_user_credential

The `orapki secretstore create_user_credential` command creates a credential object that is referenced by an alias that is constituted from a map and key name.

Syntax

```
orapki secretstore create_user_credential [-wallet wallet_file_directory] [-pwd
wallet_password]
[-map map] [-key key] [-username user_name] [-password user_password]
```

- `wallet` specifies the path to the directory where you created the wallet that will contain the user credentials. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `map` specifies the map that is used to reference a credential in the Oracle Platform Security Services (OPSS) credential store framework (CSF). This is combined with the key to construct the alias for the credential.
- `key` specifies the map that is used to reference a credential in the OPSS CSF.This is combined with the map to construct the alias for the credential.
- `username` and `password` are the credentials of the user name to be stored in the secret store. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
orapki secretstore create_user_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-map ofss.map -key cwalletkey -username pfitch
```

```
Enter wallet password: wallet_password
Enter user password: user_password
```

B.6.11 orapki secretstore delete_credential

The `orapki secretstore delete_credential` command deletes database connection credentials from a wallet.

Syntax

```
orapki secretstore delete_credential [-wallet wallet_file_directory][-pwd
wallet_password]
[-connect_string db_connect_string]
```

- `wallet` specifies the path to the wallet directory where the credential is stored. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `connect_string` can be the TNS alias that you use to specify the database in the `tnsnames.ora` file or any service name you use to identify the database on an Oracle Database network.

Example

```
orapki secretstore delete_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-connect_string sales.us.example.com
Enter wallet password: wallet_password
```

B.6.12 orapki secretstore delete_entry

The `orapki secretstore delete_entry` command deletes the secret entries for an alias from a wallet.

Syntax

```
orapki secretstore delete_entry  [-wallet wallet_file_directory] [-pwd wallet_password]
[-alias alias]
```

- `wallet` specifies the location of the wallet that contains the secret entries to be deleted for the specified alias. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `alias` specifies the name of the alias from which you want to delete the secret entries.

Example

```
orapki secretstore delete_entry -wallet $ORACLE_HOME/admin/db_unique_name/
wallet
-alias db_alias
```

B.6.13 orapki secretstore delete_user_credential

The `orapki secretstore delete_user_credential` command deletes the credential object that is referenced by the alias that was constituted from the map and key name.

Syntax

```
orapki secretstore delete_user_credential [-wallet wallet_file_directory] -pwd
wallet_password]
[-map map] [-key key]
```

- `wallet` specifies the path to the directory where you created the wallet that contains the user credentials.
- `map` specifies the map that is used to reference a credential in the Oracle Platform Security Services (OPSS) credential store framework (CSF). This is combined with the key to construct the alias for the credential.
- `key` specifies the map that is used to reference a credential in the OPSS CSF. This is combined with the key to construct the alias for the credential.

Example

```
orapki secretstore delete_user_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-map ofss.map -key cwalletkey
Enter wallet password: wallet_password
```

B.6.14 orapki secretstore list_credentials

The `orapki secretstore list_credentials` command lists the contents of the external password store.

Syntax

```
orapki secretstore list_credentials [-wallet wallet_file_directory] [-pwd
wallet_password]
```

- `wallet` specifies the location of the wallet whose external password store credentials you want to view. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.

Example

```
orapki secretstore list_credentials -wallet $ORACLE_HOME/admin/db_unique_name/
wallet
Enter wallet password: wallet_password
```

B.6.15 orapki secretstore list_entries

The `orapki secretstore list_entries` command lists the identifiers in a wallet.

The `orapki wallet display` command is a superset of the information that is shown in the `orapki secretstore list_entries` command.

Syntax

```
orapki secretstore list_entries [-wallet wallet_file_directory] [-pwd wallet_password]
```

- `wallet` specifies the location of the wallet whose identifiers you want to list. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.

Example

```
orapki secretstore list_entries -wallet $ORACLE_HOME/admin/db_unique_name/
wallet
 Enter wallet password: wallet_password
```

Related Topics

- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

B.6.16 orapki secretstore list_entries_unsorted

The `orapki secretstore list_entries_unsorted` command lists the identifiers in a wallet in unsorted order.

Syntax

```
orapki secretstore list_entries_unsorted [-wallet wallet_file_directory] [-pwd
wallet_password]
```

- `wallet` specifies the location of the wallet whose identifiers you want to list. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.

Example

```
orapki secretstore list_entries_unsorted -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
Enter wallet password: wallet_password
```

B.6.17 orapki secretstore modify_credential

The `orapki secretstore modify_credential` command modifies database connection credentials in the wallet.

Syntax

```
orapki secretstore modify_credential [-wallet wallet_file_directory]  [-pwd
[wallet_password]]
[-connect_string db_connect_string]
[-username user_name] [-password user_password]
```

- `wallet` specifies the path to the wallet directory that stores the credential. If you omit the pwd argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.

- `connect_string` is the TNS alias that you use to specify the database in the `tnsnames.ora` file or any service name you use to identify the database on an Oracle Database network.
- `username` and `password` are the database login credentials. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
orapki secretstore modify_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-connect_string sales.us.example.com -username pfitch
Enter wallet password: wallet_password
Enter user password: user_password
```

B.6.18 orapki secretstore modify_entry

The `orapki secretstore modify_entry` command modifies the secret entry for an alias in a wallet.

Syntax

```
orapki secretstore modify_entry [-wallet wallet_file_directory] [-pwd wallet_password]
[-alias alias] [-secret secret]
```

- `wallet` specifies the location of the wallet that contains the secret entriy to be modified for the specified alias. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `alias` specifies the name of the alias where the secret entries are stored.
- `secret` specifies the secret text that you store.

Example

```
orapki secretstore modify_entry -wallet $ORACLE_HOME/admin/db_unique_name/
wallet
-alias db_alias -secret Time2Cry@
Enter wallet password: wallet_password
```

B.6.19 orapki secretstore modify_user_credential

The `orapki secretstore modify_user_credential` command modifies a credential object that is referenced by an alias that was constituted from a map and key name.

Syntax

```
orapki secretstore modify_user_credential [-wallet wallet_file_directory] [-pwd
wallet_password]
[-map map] [-key key] [-username user_name] [-password user_password]
```

- `wallet` specifies the path to the directory where you created the wallet that contains the user credentials. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.

- `map` specifies the map that is used to reference a credential in the Oracle Platform Security Services (OPSS) credential store framework (CSF). This is combined with the key to construct the alias for the credential.
- `key` specifies the map that is used to reference a credential in the OPSS CSF. This is combined with the key to construct the alias for the credential.
- `username` and `password` are the credentials of the user name to be stored in the secret store. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
orapki secretstore modify_user_credential -wallet $ORACLE_HOME/admin/
db_unique_name/wallet
-map ofss.map -key cwalletkeyhr -username psmith
Enter wallet password: wallet_password
Enter user password: user_password
```

B.6.20 orapki secretstore view_entry

The `orapki secretstore view_entry` command lists the secret entries for an alias in a wallet.

Syntax

```
orapki secretstore view_entry [-wallet <wallet_file_directory>] [-pwd <wallet_password>]
[-alias <alias>]
```

- `wallet` specifies the location of the wallet that will contain the secret entries for the specified alias. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `alias` specifies the name of the alias for which the secrety entries will be displayed

Example

```
orapki secretstore view_entry -wallet $ORACLE_HOME/admin/<db_unique_name>/
wallet -alias <db_alias>
 Enter wallet password: wallet_password
```

Related Topics

- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

B.6.21 orapki wallet add

The `orapki wallet add` command adds certificate requests and certificates to an Oracle wallet.

Syntax

```
orapki wallet add [-wallet [wallet_file_directory]] [-dn [user_dn]] [-alias [alias]] -
asym_alg [RSA|ECC]
[-keysize [512|768|1024|2048|4096|8192|16384]] | [-eccurve [p192|p224|p256|p384|p521|
k163|k233|k283|k409|k571|b163|b233|b283|b409|b571]]
```

```
-self_signed [-validity [number_of_days]] | [-valid_from [mm/dd/yyyy] -valid_until
[mm/dd/yyyy]]
[-serial_file file_path] | [-serial_num serial_num] -addext_ski
-addext_ku
digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment,keyAgreement,keyCertSign
,cRLSign,encipherOnly,decipherOnly
-addext_basic_cons [CA] | [-pathLen [pathlen]]]  -addext_san [DNS:value] [-cert
[file_name]]
[-trusted_cert|-user_cert] [-pwd password] | [-auto_login_only]
[-sign_alg md5|sha1|sha256|sha384|sha512|ecdsasha1|ecdsasha256|ecdsasha384|ecdsasha512]
[-cert_validation_mode strict|non-strict]
```

Table B-2 Parameter Descriptions of orapki wallet add

Parameter	Description
`wallet`	Specifies the location of the wallet to which you want to add a certificate request.

`alias`	Specifies a unique certificate or certificate request.
	For example, it can be used to add and later export a certificate request:
	
	`orapki wallet create -wallet sample_wallet`
	`orapki wallet add -wallet sample_wallet -dn CN=ROOT -keysize 2048 -validity 365 -self_signed -alias sample_alias`
	`orapki wallet export -wallet sample_wallet -alias sample_alias -request cert_request.csr`

`dn`	Specifies the distinguished name of the certificate to add.
`keySize`	Specifies the key size in bits for the certificate. The size that you enter indicates the strength of security for the certificate. Values are as follows: • `512`: Included for backward compatibility and is supported in non-FIPS mode • `768`: Supported in non-FIPS mode • `1024`: Current default for non-FIPS certificate keys and is supported in non-FIPS mode • `2048`: Current default for FIPS certificate keys • `4096`: As needed per your site's requirements • `8192`: As needed per your site's requirements • `16384`: As needed per your site's requirements
`asym_alg`	Specifies the algorithm (`RSA` or `ECC`) to use for the certificate creation, in the case of a self-signed certificate.
`self-signed`	Creates and adds a root certificate. This option provides either the `validity` option or the `valid_from` and `valit_until` options (mandatory).
`serial_file`	Specifies the file location of the serial file for the certificate.
`serial_num`	Specifies the serial number of the certificate.
`addtext_ski`	Adds the Subject Key Identifier extension and identifies the public key certified by the certificate.
`addtext_ku <list of key usage separated by spaces>`	Adds the Key Usage extension to the certificate.

Table B-2 (Cont.) Parameter Descriptions of orapki wallet add

Parameter	Description
`addtext_basic_cons [CA] [-pathLen <pathlen>]`	Adds the Basic Constraint extension. The optional `[CA]` and `[-pathLen]` fields signify whether the given certificate is a certificate authority or not.

`addtext_san`	Is an extension to X509 certificates used to add subject alternative names, which is used in addition to identify the subject. This option only allows you to add domain names separated by comma. For example:
	
	`addext_san DNS:value_1,DNS:value_2,DNS:value_3`
	`-addext_san DNS:ns1.example.com,DNS:ns2.example.com`

`addtext_xyz`	Specifies different constraints.
`cert`	Specifies the location of certificate to add.
`trusted_cert | user_cert`	Specify the type of certificate to add, either trusted or user.
`sign_alg`	Specifies the signing algorithm to be used for signing certificates. This setting applies to self-signed certificates only.
`cert_validation_mode`	Specifies if strict certificate validation, conforming to the RFC#5280 standard is (`strict`) or is not (`non-strict`) being used.

To sign the request, export it with the export option.

To add trusted certificates:

```
orapki wallet add -wallet wallet_file_directory -trusted_cert -cert
certificate_file_directory
```

- `trusted_cert` adds the trusted certificate, at the location specified with `-cert`, to the wallet.

To add root certificates:

```
orapki wallet add -wallet wallet_file_directory -dn certificate_dn -keySize 512|1024|
2048 -self_signed -validity number_of_days
```

- `self_signed` creates a root certificate.
- `validity` is mandatory. Use it to specify the number of days, starting from the current date, that this root certificate will be valid.

To add user certificates:

```
orapki wallet add -wallet wallet_file_directory -user_cert -cert
certificate_file_directory
```

- `user_cert` adds the user certificate at the location specified with the `-cert` parameter to the wallet. Before you add a user certificate to a wallet, you must add all the trusted certificates that make up the certificate chain. If all trusted certificates are not installed in the wallet before you add the user certificate, then adding the user certificate will fail.

Example

```
orapki  wallet add -wallet $ORACLE_HOME/admin/db_unique_name/wallet -dn
"cn=mavis green, o=example, c=us" -keySize 2048
```

Related Topics

- orapki wallet export
  The `orapki wallet export` command exports certificate requests and certificates from an Oracle wallet.

B.6.22 orapki wallet change_pwd

The `orapki wallet change_pwd` command changes the password for a wallet.

Syntax

```
orapki wallet change_pwd [-wallet wallet_file_directory] [-oldpwd old_wallet_password] [-
newpwd new_wallet_password]
```

- `wallet` specifies the location of the wallet whose password you want to change.
- `oldpwd` specifies the current password to change.
- `newpwd` specifies the new password. Follow these requirements:
  - Use no fewer than 8 characters. The maximum length is unlimited.
  - Use mixed alphanumeric characters.

Example

```
orapki wallet change_pwd -wallet wallet_file_directory -oldpwd
old_wallet_password -newpwd new_wallet_password
Enter password: wallet_password
```

B.6.23 orapki wallet convert

The `orapki wallet convert` command converts the 3DES algorithm in an Oracle wallet to use the AES256 algorithm.

Be aware that though the AES256 algorithm is stronger than 3DES, there will be degradation in `orapki` operations if you use AES256.

Syntax

```
orapki wallet convert -wallet wallet_file_directory [-pwd wallet_password] -compat_v12
```

- `wallet` specifies the wallet location for which you want to turn on auto-login.
- `pwd` is the wallet password. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.
- `compat_v12` performs the conversion from 3DES to AES256.

Example

```
orapki  wallet convert -wallet $ORACLE_HOME/admin/db_unique_name/wallet
compat_v12
Enter wallet password: password
```

B.6.24 orapki wallet create

The `orapki wallet create` command creates an Oracle wallet or enables auto-login for an Oracle wallet.

Syntax

```
orapki wallet create [-wallet wallet_file_directory] [-pwd wallet_password] [-
auto_login|-auto_login_local]] | [-auto_login_only]
```

- `wallet` specifies a location for the new wallet or the location of the wallet for which you want to turn on auto-login.
- `pwd` is a new password to be assigned to the wallet. If you create an auto-login wallet later on, then it will require this password. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted. When you create the password, follow these requirements:
  - Use no fewer than 8 characters. The maximum length is unlimited.
  - Use mixed alphanumeric characters.
- `auto_login` creates an auto-login wallet, or it turns on automatic login for the wallet specified with the `-wallet` option.
- `auto_login_only` is a type of auto-login wallet that does not require a password.
- `auto_login_local` creates a local auto-login wallet, or it turns on local automatic login for the wallet specified with the `-wallet` option.

Example

```
orapki wallet create -wallet $ORACLE_HOME/admin/db_unique_name/wallet
Enter password: wallet_password
Enter password again: password
```

B.6.25 orapki wallet delete

The `orapki wallet delete` command deletes an Oracle wallet.

Syntax

```
orapki wallet delete [-wallet wallet_file_directory] [-pwd wallet_password] [-sso]
```

- `wallet` specifies the location of the wallet that you want to delete. If you omit the `pwd` argument for the password, then you will be prompted for the password. For better security, enter the password when prompted.
- `sso` enables you to delete an auto-login wallet.

Example

```
orapki wallet delete -wallet $ORACLE_HOME/admin/db_unique_name/wallet -sso
Enter password: wallet_password
```

B.6.26 orapki wallet display

The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

The `orapki wallet display` command is a superset of the information that is shown in the `orapki secretstore list_entries` command. `orapki wallet display` shows everything, including the secret store entries' thumbprint. It inlcudes both the SHA-1 and SHA-256 thumbprint information for a private key. These thumbprints select a particular certificate from the wallet and are displayed when you run the `orapki wallet display` command. You can specify an alias when you store a private key. The alias and thumbprint enable you to specify the exact private key to use with the connect string.

Syntax

```
orapki wallet display [-wallet [wallet_file_directory]] [-summary | [-complete | -
complete -details]]
[-pwd wallet_password] [-ssvs]
```

- `wallet` specifies a location for the wallet you want to open if it is not located in the current working directory.
- `summary` displays a summary of the wallet information; `complete` displays more details.
- `ssvs` displays the version of the wallet.
- `details` displays additional attributes such as version, signature algorithm, subject public key information, and extensions, as follows:
  - `summary` is the subject name.
  - `complete` contains `Alias`, `Subject`, `Issuer`, `Not Before`, `Not After`, `Serial Number`, `Key Length`, `MD5 digest`, `SHA-256 digest`, `SHA-1 digest`, `Thumbprint`
  - `details` contains `Alias`, `Subject`, `Version`, `Subject`, `Issuer`, `Serial Number`, `Not Before`, `Not After`, `Fingerprint`, `Signature Algorithm`, `MD5 digest`, `SHA-256 digest (thumbprint)`, `SHA-1 digest (thumbprint)`, `Subject Public Key Information` (which includes `Key Algorithm`, `Key Length`, and `Key Data`), and, if any, `Certificate Extensions`.

Example

```
orapki  wallet display -wallet $ORACLE_HOME/admin/db_unique_name/wallet
```

Related Topics

- orapki secretstore list_entries
  The `orapki secretstore list_entries` command lists the identifiers in a wallet.

B.6.27 orapki wallet export

The `orapki wallet export` command exports certificate requests and certificates from an Oracle wallet.

Syntax

```
orapki wallet export -wallet wallet_file_directory -dn certificate_dn -cert
certificate_filename
```

- `wallet` specifies the location of the wallet from which you want to export the certificate.
- `dn` specifies the distinguished name of the certificate. In the case of a multi-valued DN, the order in which the individual DN values are stored in the wallet is uncertain. To find the correct DN that you want, run `orapki wallet display`.
- `cert` specifies the name of the file that contains the exported certificate.

To export a certificate request from an Oracle wallet:

```
orapki wallet export -wallet ./rsa_server_host_name -dn "O=Example, C=US" -request ./
rsa_server_hostname/csr2.pem
Enter wallet password: password
```

- `request` specifies the name of the file that contains the exported certificate request.

Example

```
orapki  wallet export -wallet $ORACLE_HOME/admin/db_unique_name/wallet
-dn db_cert
-request db_req
```

Related Topics

- orapki wallet display
  The `orapki wallet display` command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

B.6.28 orapki wallet export_private_key

The `orapki wallet export_private_key` command exports a private key from a wallet.

Syntax

```
orapki wallet export_private_key [-wallet wallet_file_directory] [-pwd wallet_password]
[-alias pvtkey_alias]
[-pvtkeyfile filename] [-pvtkeypwd private_key_password] [-salt salt]
[-cert certificate_filename] [-cacert ca_certificate_filename]
```

- `wallet` specifies the location of the wallet from which you want to export the private key.
- `pvtkeyfile` specifies the name of the private key file
- `pvtkeypwd` specifies password for the private key file. If omitted, a password prompt appears.
- `salt` specifies the salt to use.
- `cert` specifies certificate file name.

- `cacert` specifies the CA file name .

Example

```
orapki wallet export_private_key -wallet wallet_file_directory -alias
pvtkey_alias
-pvtkeyfile pvt_key_filename -pvtkeypwd pvt_key_password -cert cert_file -
cacert cacert_file
Enter password: wallet_password
```

B.6.29 orapki wallet import_pkcs12

The `orapki wallet import_pkcs12` command imports a PKCS #12 file into the wallet. Only the latest valid certificate for each unique private key in a PKCS#12 file will be imported into an Oracle wallet. If a private key already exists in the wallet, its associated certificate chain will be skipped.

Syntax

```
orapki wallet import_pkcs12 -wallet wallet_location [-pwd wallet_password]
[-auto_login_only]] -pkcs12file pkcs12_file_location [-pkcs12pwd pkcs12_file_password]
```

- `wallet` specifies the location into which PKCS#12 file is to be imported..
- `pkcs12file` specifies the location of the PKCS#12 file to be imported into the wallet.
- `pkcs12pwd` specifies the password of PKCS#12 file that is to be imported into the wallet. If omitted, a password prompt appears.

Example

```
orapki wallet import_pkcs12 -wallet wallet_location -pkcs12file
pkcs12_file_location -pkcs12pwd pkcs12_file_password
Enter password: wallet_password
```

B.6.30 orapki wallet import_private_key

The `orapki wallet import_private_key` command imports a private key into a wallet.

Syntax

```
orapki wallet import_private_key [-wallet wallet_file_directory] [-pwd wallet_password]
[-alias pvtkey_alias]
[-pvtkeyfile filename] [-pvtkeypwd private_key_password] [-salt salt]
[-cert certificate_filename] [-cacert ca_certificate_filename] [-cert_validation_mode
strict|non-strict]
```

- `wallet` specifies the location of the wallet into which you want to import the private key.
- `pvtkeyfile` specifies the name of the private key file
- `pvtkeypwd` specifies password for the private key file. If omitted, a password prompt appears.
- `salt` specifies the type of salt to use.
- `cert` specifies certificate file name.

- `cacert` specifies the CA file name.
- `cert_validation_mode` specifies if strict certificate validation, conforming to the RFC#5280 standard is (`strict`) or is not (`non-strict`) being used.

Example

```
orapki wallet import_private_key -wallet wallet_file_directory -alias
pvtkey_alias
-pvtkeyfile pvt_key_filename -pvtkeypwd pvt_key_password -cert cert_file -
cacert cacert_file
Enter password: wallet_password
```

B.6.31 orapki wallet jks_to_pkcs12

The `orapki wallet jks_to_pkcs12` command converts a Java keystore to PKCS #12 format for the storage of certificate information.

To convert a wallet that uses PKCS #12 format to a Java keystore, you can use `orapki wallet pkcs12_to_jks` command.

Syntax

```
orapki wallet jks_to_pkcs12 [-wallet wallet_file_directory] [-pwd wallet_password]
[-keystore keystore] [-jkspwd jks_password]
```

- `wallet` specifies the location of the wallet that you want to convert to use PKCS #12 format.
- `keystore` specifies the name of the Java keystore to convert.
- `jkspwd` specifies the password of the Java keystore. If omited, a password prompt appears.

Example

```
orapki wallet jks_to_pkcs12 -wallet wallet_file_directory -keystore
keystore_name -jkspwd keystore_password
Enter password: wallet_password
```

B.6.32 orapki wallet pkcs12_to_jks

The `orapki wallet pkcs12_to_jks` command converts a PKCS #12 keystore to a Java keystore for the storage of certificate information.

To convert a Java keystore wallet to PKCS #12 format to a Java keystore, you can use `orapki wallet jks_to_pkcs12` command.

Syntax

```
orapki wallet pkcs12_to_jks [-wallet wallet_file_directory] [-pwd wallet_password]
[-jksKeyStoreLoc Java_keystore_location -jksKeyStorepwd Java_keystore_password]
[-jksTrustStoreLoc jks_trust_store_location -jksTrustStorepwd jks_trust_store_password]
```

- `wallet` specifies the location of the wallet that you want to convert to use Java keystore format.

- `jksKeyStoreLoc` specifies the location for the Java keystore that will be created.
- `jksTrustStorepwd` specifies the password of the JKS trust store. If omitted, a password prompt appears.

Example

```
orapki wallet pkcs12_to_jks -wallet wallet_file_directory -jksKeyStoreLoc
Java_keystore_location -jkspwd Java_keystore_password
Enter password: wallet_password
```

B.6.33 orapki wallet remove

The `orapki wallet remove` command removes certificates and certificate requests from the wallet.

Syntax

```
orapki wallet remove [-wallet wallet_file_directory] [-dn subject_dn] | -alias alias]
[-issuer_dn issuer_dn] [-serial_file file_path] | [-serial_num serial_num]
[-trusted_cert_all|-trusted_cert|-user_cert|-cert_req] [-pwd wallet_password | [-
auto_login_only]
```

- `wallet` specifies the location of the file where a certificate or certificate request will be removed.
- `dn` specifies distinguished name of the wallet.
- `alias` specifies the alias for this wallet.
- `issuer_dn` specifies the issuer of the DN.
- `trusted_cert_all|-trusted_cert|-user_cert|-cert_req` specifies the type of certificate to remove from the wallet.
- `serial_file` specifies the file location of the serial file for the certificate.
- `serial_num` specifies the serial number of the certificate.

Example

```
orapki wallet remove -wallet wallet_file_directory -dn certificate_dn
Enter password: wallet_password
```

B.7 mkstore Utility Commands Summary

The `mkstore` command line utility, available as part other Oracle Database client and server installations, enables you to create wallets and add credential secrets such as user names and passwords.

Starting with Oracle Database release 23ai, `mkstore` is deprecated. Use `orapki` instead.

- mkstore create
  The `mkstore create` command creates a wallet (`cwallet.sso` and `ewallet.p12`) at the command line.
- mkstore createALO
  The `mkstore createALO` command creates an auto-login-only wallet (`cwallet.sso`).

- mkstore createCredential
  The `mkstore createCredential` command creates database connection credentials in the wallet.
- mkstore createEntry
  The `mkstore createEntry` command stores a secret text against an alias.
- mkstore createUserCredential
  The `mkstore createUserCredential` command creates a credential object that is referenced by an alias that is constituted from a map and key name.
- mkstore delete
  The `mkstore delete` command deletes a wallet.
- mkstore deleteCredential
  The `mkstore deleteCredential` command deletes database login credentials from a wallet.
- mkstore deleteEntry
  The `mkstore deleteEntry` command deletes the secret entries for an alias in a wallet.
- mkstore deleteSSO
  The `mkstore deleteSSO` command deletes an auto-login wallet.
- mkstore deleteUserCredential
  The `mkstore deleteUserCredential` command deletes the credential object that is referenced by the alias that was constituted from the map and key name.
- mkstore list
  The `mkstore list` command lists the identifiers in a wallet.
- mkstore listCredential
  The `mkstore listCredential` command lists the contents of the external password store.
- mkstore modifyCredential
  The `mkstore modifyCredential` command modifies the database login credentials that are in a wallet.
- mkstore modifyEntry
  The `mkstore modifyEntry` command modifies the secret entries for an alias in a wallet.
- mkstore modifyUserCredential
  The `mkstore modifyUserCredential` command modifies a credential object that is referenced by an alias constituted from a map and key name.
- mkstore viewEntry
  The `mkstore viewEntry` command lists the secret entries for an alias in a wallet.

B.7.1 mkstore create

The `mkstore create` command creates a wallet (`cwallet.sso` and `ewallet.p12`) at the command line.

Syntax

```
mkstore -wrl wallet_file_directory -create
```

- `wrl` specifies the path to the directory where you want to create and store the wallet.
- This command prompts you to enter and reenter a new password. When you create the password, follow these requirements:
  - Use no fewer than 8 characters. The maximum length is unlimited.

– Use mixed alphanumeric characters.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -create
Enter password: password
Enter password again: password
```

Related Topics

- orapki wallet create
  The `orapki wallet create` command creates an Oracle wallet or enables auto-login for an Oracle wallet.

B.7.2 mkstore createALO

The `mkstore createALO` command creates an auto-login-only wallet (`cwallet.sso`).

Syntax

```
mkstore -wrl wallet_file_directory -createALO
```

- `wrl` specifies the path to the directory where you want to create and store the auto-login-only wallet.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -createALO
```

Related Topics

- orapki wallet create
  The `orapki wallet create` command creates an Oracle wallet or enables auto-login for an Oracle wallet.

B.7.3 mkstore createCredential

The `mkstore createCredential` command creates database connection credentials in the wallet.

Syntax

```
mkstore -wrl wallet_file_directory -createCredential db_connect_string username password
```

- `wrl` specifies the path to the directory where you created the wallet.
- *db_connect_string* can be the TNS alias that you use to specify the database in the `tnsnames.ora` file or any service name you use to identify the database on an Oracle Database network.
- *username* and *password* are the database login credentials. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -createCredential DBFS
dbfs_admin
Enter password: password
```

Related Topics

- orapki secretstore create_credential
  The `orapki secretstore create_credential` command creates database connection credentials in the wallet.

B.7.4 mkstore createEntry

The `mkstore createEntry` command stores a secret text against an alias.

Syntax

```
mkstore -wrl wallet_file_directory -createEntry alias secret
```

- `wrl` specifies the path to the directory wallet for which you want to create the entry.
- *`alias`* is the name of the alias for which you want to store the secret text.
- *`secret`* specifies the secret text that you want to store.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -createEntry
oracle.security.client.default_username SCOTT
```

Related Topics

- orapki secretstore create_entry
  The `orapki secretstore create_entry` command stores a secret entries against an alias in a wallet.

B.7.5 mkstore createUserCredential

The `mkstore createUserCredential` command creates a credential object that is referenced by an alias that is constituted from a map and key name.

Syntax

```
mkstore -wrl wallet_file_directory  -createUserCredential map key username password
```

- `wrl` specifies the path to the directory where you created the wallet.
- *`map`* is the map that is used to reference a credential in the Oracle Platform Security Services (OPSS) credential store framework (CSF). This is combined with the key to construct the alias for the credential.
- *`key`* is the key used to reference a credential in the OPSS CSF. This is combined with the map to construct the alias for the credential.
- *`username`* is the user name to be stored in the secret store. If a user name is not specified, then `mkstore` sets it as `NO_USER` in the credential.

- *password* is the password to be stored in the secret store. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -createUserCredential
ofss.map cwalletkey ofss
Enter your secret/Password: password
Re-enter your secret/Password: password
```

Related Topics

- orapki secretstore create_user_credential
  The `orapki secretstore create_user_credential` command creates a credential object that is referenced by an alias that is constituted from a map and key name.

B.7.6 mkstore delete

The `mkstore delete` command deletes a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -delete
```

- `wallet` specifies the location of the wallet to be deleted.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -delete
Enter wallet password: password
```

Related Topics

- orapki wallet delete
  The `orapki wallet delete` command deletes an Oracle wallet.

B.7.7 mkstore deleteCredential

The `mkstore deleteCredential` command deletes database login credentials from a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -deleteCredential connect_string
```

- `wrl` specifies the location of the wallet that contains the credentials to be deleted.
- *connect_string* can be the TNS alias you use to specify the database in the `tnsnames.ora` file, or any service name that you use to identify the database on an Oracle Database network.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -deleteCredential DBFS
dbfs_admin
Enter wallet password: password
```

Related Topics

- orapki secretstore delete_credential
  The `orapki secretstore delete_credential` command deletes database connection credentials from a wallet.

B.7.8 mkstore deleteEntry

The `mkstore deleteEntry` command deletes the secret entries for an alias in a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -deleteEntry alias
```

- `wrl` specifies the location of the wallet that contains the secret entries to be deleted for the specified alias.
- *`alias`* specifies the name of alias for which you want to delete the secret entries.
- This command prompts you to enter and reenter a new password. When you create the password, follow these requirements:
  - Use no fewer than 8 characters. The maximum length is unlimited.
  - Use mixed alphanumeric characters.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -deleteEntry db_alias
Enter wallet password: password
```

Related Topics

- orapki secretstore delete_entry
  The `orapki secretstore delete_entry` command deletes the secret entries for an alias from a wallet.

B.7.9 mkstore deleteSSO

The `mkstore deleteSSO` command deletes an auto-login wallet.

Syntax

```
mkstore -wrl wallet_file_directory -deleteSSO
```

- `wrl` specifies the location of the SSO wallet to delete.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -deleteSSO
Enter wallet password: password
```

Related Topics

- orapki wallet delete
  The `orapki wallet delete` command deletes an Oracle wallet.

B.7.10 mkstore deleteUserCredential

The `mkstore deleteUserCredential` command deletes the credential object that is referenced by the alias that was constituted from the map and key name.

Syntax

```
mkstore -wrl wallet_file_directory -deleteUserCredential map key
```

- `wrl` specifies the location of the wallet that contains the credential object to delete.
- *map* specifies the map that used to reference a credential in the Oracle Platform Security Services (OPSS) credential store framework (CSF). This is combined with the key to construct the alias for the credential.
- *key* specifies the key that used to reference a credential in the OPSS CSF. This is combined with the map to construct the alias for the credential.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -deleteUserCredential
ofss.map cwalletkey
Enter wallet password: password
```

Related Topics

- orapki secretstore delete_user_credential
  The `orapki secretstore delete_user_credential` command deletes the credential object that is referenced by the alias that was constituted from the map and key name.

B.7.11 mkstore list

The `mkstore list` command lists the identifiers in a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -list
```

- `wrl` specifies the location of the wallet whose identifiers you want to list.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -list
Enter wallet password: password
```

Related Topics

- orapki secretstore list_entries
  The `orapki secretstore list_entries` command lists the identifiers in a wallet.

B.7.12 mkstore listCredential

The `mkstore listCredential` command lists the contents of the external password store.

Syntax

```
mkstore -wrl wallet_file_directory -listCredential
```

- `wrl` specifies the location of the wallet whose external password store credentials you want to view.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -listCredential
Enter wallet password: password
```

Related Topics

- orapki secretstore list_credentials
  The `orapki secretstore list_credentials` command lists the contents of the external password store.

B.7.13 mkstore modifyCredential

The `mkstore modifyCredential` command modifies the database login credentials that are in a wallet.

Syntax

```
mkstore -wrl wallet_file_directory] -modifyCredential connect_string username password
```

- `wrl` specifies the location of the wallet.
- *`db_connect_string`* can be the TNS alias that you used to specify the database in the `tnsnames.ora` file or the service name you used to identify the database on an Oracle Database network.
- *`username`* and *`password`* are the database login credentials. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -modifyCredential DBFS
sec_admin
Enter your secret/Password: password
Re-enter your secret/Password: password
```

Related Topics

- orapki secretstore modify_credential
  The `orapki secretstore modify_credential` command modifies database connection credentials in the wallet.

B.7.14 mkstore modifyEntry

The `mkstore modifyEntry` command modifies the secret entries for an alias in a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -modifyEntry alias secret
```

- `wrl` specifies the location of the wallet that contains the secret entries to modify.
- *`alias`* is the name of the alias for the secret text.
- *`secret`* specifies the secret text.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -modifyEntry
oracle.security.client.default_username PSMITH
Enter wallet password: password
```

Related Topics

- orapki secretstore modify_entry
  The `orapki secretstore modify_entry` command modifies the secret entry for an alias in a wallet.

B.7.15 mkstore modifyUserCredential

The `mkstore modifyUserCredential` command modifies a credential object that is referenced by an alias constituted from a map and key name.

Syntax

```
mkstore -wrl wallet_file_directory -modifyUserCredential map key username password
```

- `wallet` specifies the location of the wallet whose user credentials need to be modified.
- *`map`* is an attribute that is used to reference a credential. This is combined with the key to construct the alias for the credential.
- *`key`* is the key used to reference a credential. This is combined with the map to construct the alias for the credential.

- *username* is the user name to be stored in the secret store. If a user name is not specified, then `mkstore` sets it as `NO_USER` in the credential.
- *password* is the password to be stored in the secret store. If no password is provided, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -modifyUserCredential
connect_string.map cwalletkey sample_user
Enter your secret/Password: password
Re-enter your secret/Password: password
Enter wallet password: password
```

Related Topics

- orapki secretstore modify_user_credential
  The `orapki secretstore modify_user_credential` command modifies a credential object that is referenced by an alias that was constituted from a map and key name.

B.7.16 mkstore viewEntry

The `mkstore viewEntry` command lists the secret entries for an alias in a wallet.

Syntax

```
mkstore -wrl wallet_file_directory -viewEntry alias
```

- `wrl` specifies the location of the wallet that contains the secret entries to view.
- *alias* specifies the name of alias.
- This command prompts you to enter the wallet password.

Example

```
mkstore -wrl $ORACLE_HOME/admin/db_unique_name/wallet -viewEntry db_alias
Enter wallet password: password
```

Related Topics

- orapki secretstore view_entry
  The `orapki secretstore view_entry` command lists the secret entries for an alias in a wallet.