# Using The Secure External Password Store (Doc ID 340559.1)

**In this Document**

## APPLIES TO:

Oracle Database - Standard Edition - Version 10.2.0.5 and later
Oracle Database - Enterprise Edition - Version 10.2.0.5 and later
Oracle Database Exadata Cloud Machine - Version N/A and later
Oracle Cloud Infrastructure - Database Service - Version N/A and later
Oracle Database Cloud Exadata Service - Version N/A and later
Information in this document applies to any platform.

## GOAL

Password credentials for connecting to databases can now be stored in a client-side Oracle wallet, a secure software container used to store authentication and signing credentials. This wallet usage can simplify large-scale deployments that rely on password credentials for connecting to databases. When this feature is configured, application code, batch jobs, and scripts no longer need embedded user names and passwords. Risk is reduced because such passwords are no longer exposed in the clear, and password management policies are more easily enforced without changing application code whenever user names or passwords change.

The feature Secure External Password Store can be used without any restriction in all product editions, you do not require a license for the Advanced Security Option (ASO).

## SOLUTION

When clients are configured to use the secure external password store, applications can connect to a database with the following `CONNECT` statement syntax, without specifying database login credentials:
connect /@db_connect_string

where db_connect_string is a valid connect string to access the intended database. In this case, the database credentials, username and password, are securely stored in an Oracle wallet created for this purpose. The autologin feature of this wallet is turned on so the system does not need a password to open the wallet. From the wallet, it gets the credentials to access the database for the user they represent.

**Configuring Clients to Use the External Password Store**

1) Create a wallet on the client by using the following syntax at the command line:

```
mkstore -wrl <wallet_location> -create

example:

mkstore -wrl /home/mseibt/pstore -create
Enter password: welcome1
Enter password again: welcome1

ls -al /home/mseibt/pstore
```

```
-rw------- 1 mseibt dba 7940 Nov 9 15:38 cwallet.sso
-rw------- 1 mseibt dba 7912 Nov 9 15:38 ewallet.p12
```

2) Create database connection credentials in the wallet by using the following syntax at the command line:

```
mkstore -wrl <wallet_location> -createCredential <db_connect_string> <username> <password>

example:

("N102" in the following example is a connect descriptor located in the tnsnames.ora.)

mkstore -wrl /home/mseibt/pstore -createCredential N102 <user> <password>
Enter password: welcome1

Create credential oracle.security.client.connect_string1
```

Enclose usernames and passwords that have special characters in single quotes.

example:

mkstore -wrl /home/mseibt/pstore -createCredential N102 '<user>' <password>

3) In the client sqlnet.ora file, enter the WALLET_LOCATION parameter and set it to the directory location of the w

```
WALLET_LOCATION =
   (SOURCE =
      (METHOD = FILE)
      (METHOD_DATA = (DIRECTORY = /home/mseibt/pstore))
)
```

4) In the client sqlnet.ora file, enter the SQLNET.WALLET_OVERRIDE parameter and set it to TRUE

```
SQLNET.WALLET_OVERRIDE = TRUE
```

This setting causes all CONNECT /@db_connect_string statements to use the information in the
wallet at the specified location to authenticate to databases.

When external authentication is in use, an authenticated user with such a wallet can use the CONNECT /@db_connect_string syntax to access the previously specified databases without providing a user name and password. However, if a user fails that external authentication, then these connect statements will also fail.

If an application uses SSL for encryption, then the sqlnet.ora parameter, SQLNET.AUTHENTICATION_SERVICES, specifies SSL and an SSL wallet is created. If this application wants to use secret store credentials to authenticate to databases (instead of the SSL certificate), then those credentials must be stored in the SSL wallet. After SSL authentication, if SQLNET.WALLET_OVERRIDE = TRUE, then the user names and passwords from the wallet are used to authenticate. If SQLNET.WALLET_OVERRIDE = FALSE the SSL certificate is used.

Configured sqlnet.ora.

```
WALLET_LOCATION =
    (SOURCE =
       (METHOD = FILE)
       (METHOD_DATA = (DIRECTORY = /home/mseibt/pstore))
)

SQLNET.WALLET_OVERRIDE = TRUE
```

5) With the external password store configured, connect as <user>:

```
sqlplus /@N102
```

```
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Nov 9 15:59:42 2005
Copyright (c) 1982, 2005, Oracle. All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options

SQL> show user
USER is "<user>"
```

**Managing External Password Store Credentials**

1) Listing the contents of the external password store:

```
mkstore -wrl /home/mseibt/pstore -listCredential
Enter password:

List credential (index: connect_string username)
1: N102 <user>
```

2) Adding database login credentials to an existing client wallet:

```
mkstore -wrl /home/mseibt/pstore -createCredential N101 <user><password>
Enter password:

Create credential oracle.security.client.connect_string2

mkstore -wrl /home/mseibt/pstore -listCredential
Enter password:

List credential (index: connect_string username)
1: N102 <user>
2: N101 <user>
```

3) Modifying database login credentials in a wallet:

```
mkstore -wrl /home/mseibt/pstore -modifyCredential N102 <user> newpassword
Enter password:

Modify credential
Modify 1
```

4) Deleting database login credentials from a wallet:

```
mkstore -wrl /home/mseibt/pstore -deleteCredential N101
Enter password:

Delete credential
Delete 2
```

5) Listing wallet entries:

```
mkstore -wrl /home/mseibt/pstore -list
Enter password:

Oracle Secret Store entries:
oracle.security.client.connect_string1
oracle.security.client.password1
oracle.security.client.username1
```

6) Listing entry values:

```
mkstore -wrl /home/mseibt/pstore -viewEntry oracle.security.client.connect_string1
Enter password:

oracle.security.client.connect_string1 = N102

mkstore -wrl /home/mseibt/pstore -viewEntry oracle.security.client.username1
Enter password:

oracle.security.client.username1 = user
```

```
mkstore -wrl /home/mseibt/pstore -viewEntry oracle.security.client.password1
Enter password:

oracle.security.client.password1 = <password>
```

7) Modifying entry values:

```
mkstore -wrl /home/mseibt/pstore -modifyEntry oracle.security.client.password1 newpass
Enter password:

mkstore -wrl /home/mseibt/pstore -viewEntry oracle.security.client.password1
Enter password:

oracle.security.client.password1 = newpass
```

## REFERENCES

NOTE:1441745.1 - Using a Secure External Password Store with the JDBC Thin Driver
NOTE:1556219.1 - Orapki -auto_login_local Generates ORA-12578 with Secure External Password Store