25

Configuring PKI Certificate Authentication

You can configure Oracle Database to use PKI certificates for end-user authentication.

- How Oracle Database Uses Transport Layer Security for Authentication
  Transport Layer Security works with the core Oracle Database features such as encryption and data access controls.
- Enabling Oracle Internet Directory to Use Transport Layer Security Authentication
  To enable Oracle Internet Directory (OID) to use Transport Layer Security (TLS), create a wallet and certificates, and modify `tnsnames.ora` and `sqlnet.ora`.
- Configuring User Authentication with Transport Layer Security
  Both the client and server side can authenticate administrative users with Transport Layer Security (TLS).
- Configuring Transport Layer Security for Client Authentication and Encryption with X.509 Certificates
  You must perform this type of configuration on the server first, then the client.
- Configuring Email over Transport Layer Security with an Oracle Wallet
  You can use an Oracle wallet, PL/SQL packages, and security access control lists (ACLs) to configure email over a Transport Layer Security (TLS) connection.
- Troubleshooting Transport Layer Security Errors
  Oracle provides a utility to help troubleshoot PKI certificate configurations as well as additional guidance below. A utility is available through the support website to review and provide feedback on your PKI certificate authentication client and server configuration.

25.1 How Oracle Database Uses Transport Layer Security for Authentication

Transport Layer Security works with the core Oracle Database features such as encryption and data access controls.

By using Oracle Database TLS functionality to secure communications between clients and servers, you can

- Use TLS to encrypt the connection between clients and servers
- Authenticate any client or server, such as Oracle Application Server 10g, to any Oracle database server that is configured to communicate over TLS

You can use TLS features by themselves or in combination with other authentication methods supported by Oracle Database. For example, you can use the encryption provided by TLS in combination with the authentication provided by Kerberos. TLS supports any of the following authentication modes:

- Only the server authenticates itself to the client
- Both client and server authenticate themselves to each other

25.2 Enabling Oracle Internet Directory to Use Transport Layer Security Authentication

To enable Oracle Internet Directory (OID) to use Transport Layer Security (TLS), create a wallet and certificates, and modify `tnsnames.ora` and `sqlnet.ora`.

1. Log in to the database client server that has Oracle Internet Directory (OID) installed.
2. Go to the `$ORACLE_HOME/ldap/lib` directory
3. Run the following command:

   ```
   make -f ins_ldap.mk install
   ```

4. Go to the directory where the OID `tnsnames.ora` file is located.

   By default, this directory is `$ORACLE_HOME/network/admin`.

5. Edit the `tnsnames.ora` file to include the following OID settings, which will specify the TCPS port.

   For example:

   ```
   OIDDB=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)
      (HOST=sales_db.us.example.com)(PORT=5500))
       (CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=orcl.us.example.com)))
        (SECURITY=(SSL_SERVER_CERT_DN="CN=Server,O=Example,ST=California,C=US"))
   ```

   In this example, `SSL_SERVER_CERT_DN` points to the DN of the database server certificate.

6. Configure the wallet location in the `sqlnet.ora` file.

   For example:

   ```
   ENCRYPTION_WALLET_LOCATION=
    (SOURCE=
     (METHOD=FILE)
      (METHOD_DATA=
       (DIRECTORY=/etc/ORACLE/WALLETS/$ORACLE_SID/)))
   ```

7. Ensure that the `sqlnet.ora` file has the following settings:

   ```
   SSL_CLIENT_AUTHENTICATION = FALSE
   SSL_SERVER_DN_MATCH=OFF
   ```

8. Use the `orapki` utility to create a new wallet and add database certificates to it.

   For example:

   ```
   orapki wallet create -wallet /etc/ORACLE/WALLETS/$ORACLE_SID/oid_wallet
   -auto_login -pwd wallet_password
   orapki wallet add -wallet /etc/ORACLE/WALLETS/$ORACLE_SID/oid_wallet
   -trusted_cert -cert /etc/ORACLE/certificates/dbssl/root/b64certificate.txt
   -pwd wallet_password
   ./orapki wallet add -wallet /etc/ORACLE/WALLETS/$ORACLE_SID/oid_gwallet
   -trusted_cert -cert /etc/ORACLE/certificates/dbssl/netadmin/cert.txt -pwd
   wallet_password
   ```

25.3 Configuring User Authentication with Transport Layer Security

Both the client and server side can authenticate administrative users with Transport Layer Security (TLS).

1. The client needs to specify use of the PKI certificate to authenticate the end-user. If all the client connections will use this authentication method, then set `AUTHENTICATION_SERVICES=(tcps)`.

   Alternatively, you can set it separately for each connection by using `AUTHENTICATION_SERVICE=tcps` in the connect string.

   > **Note:**
   >
   > The connection string parameter is singular, while the sqlnet.ora parameter is plural.

2. For both the client and the server, ensure that the wallet has Certificate Authority (CA) certificates for user's certificate and the server's certificates. These CA certifcates can be different on the client and server.
3. Configure the client to use TLS:
   a. Add the signed user certificate to the client wallet. The CA root trust certificate should already be in the client wallet. Ensure that any intermediate certificates that are required for the user certificate are added to the wallet before you add the user certificate.

      You can use `orapki` to configure the client wallet and user certificate.

   b. Set TLS as an authentication service in the `sqlnet.ora` file.

      ```
      SSL_CLIENT_AUTHENTICATION=TRUE
      ```

   c. Optionally, for better security, set the client to use full or partial DN matching.

      When DN matching is enabled, the client will check the server certificate to ensure that host names will match what the client is configured to match. You perform this step when you enable Oracle Internet Directory to use TLS.

      > **Note:**
      >
      > The database client and server will use the strongest TLS protocol and cipher suite to establish a connection. Therefore, you do not need to specify the TLS version and cipher suites unless you have specific security requirements that require it. Be aware that if you set specific TLS versions and cipher suites, you will need to update the configuration when the older versions are no longer used.

4. Configure the listener for TLS.
   a. Create a separate listener entry for TLS connections using the secure database port 1522.

For example:

```
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = example.com)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCPS)(HOST = example.com)(PORT = 1522))
    )
  )
```

b. Comment out the non-TLS listener entry (for example, the line with `PROTOCOL = TCP`) or leave it in for non-TLS required connections.

The same wallet that the server uses can be used by the listener, along with the same server certificate. The listener will look for the wallet using the standard Oracle Database wallet search order. Alternatively, you can specify the wallet location in the listener by setting the `WALLET_LOCATION` parameter. (You cannot use the `WALLET_ROOT` parameter for this purpose, because the listener cannot use it.)

5. Configure the server to use TLS:

a. For the TLS server wallet, do the following:

- Set the `WALLET_ROOT` parameter to a location for the TLS server.
- Create the `tls` directory under *`WALLET_ROOT`*`/`*`pdb_guid`*.
- Move the TLS server wallet to the *`WALLET_ROOT`*`/`*`pdb_guid`*`/tls` directory.

b. In the `sqlnet.ora` file, add the following parameter:

```
SSL_CLIENT_AUTHENTICATION=TRUE
```

If you want to restrict authentication to only TCPS, then set `AUTHENTICATION_SERVICES` to `TCPS`.

6. Create a new schema or alter an existing schema to map to the user.

```
CREATE USER user_name IDENTIFIED EXTERNALLY AS 'user DN on certificate';
```

7. Grant the database schema to appropriate administrative privileges, such as `SYSDBA`, `SYSOPER`, and so on.

Administrative users with TLS authentication can authenticate with TLS. To enable these users, grant the appropriate administrative privilege to the user schema. The administrative user must log in using this administrative privilege. For example, for a user who was granted the `SYSOPER` administrative privilege:

```
CONNNECT /@pdb_name AS SYSOPER
```

Afterward, this user can log in by including the net service name in the `CONNECT` statement in SQL*Plus. For example, to log on as `SYSDBA` if the net service name is `orcl`:

```
CONNECT /@orcl AS SYSDBA
```

Related Topics

- 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.

- Enabling Oracle Internet Directory to Use Transport Layer Security Authentication
  To enable Oracle Internet Directory (OID) to use Transport Layer Security (TLS), create a wallet and certificates, and modify `tnsnames.ora` and `sqlnet.ora`.
- 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).

25.4 Configuring Transport Layer Security for Client Authentication and Encryption with X.509 Certificates

You must perform this type of configuration on the server first, then the client.

- About Configuring TLS for Client Authentication and Encryption with X.509 Certificates
  You can enable Public Key Infrastructure (PKI) authentication between Oracle Database clients and an Oracle database with X.509 certificates.
- Configuring the Server for Authentication and Encryption with X.509 Certificates
  You must configure the server's `listener.ora`, `sqlnet.ora`, and initialization files and create a database user account for authentication and encryption with X.509 certificates.
- Configuring the Client for Authentication and Encryption with X.509 Certificates
  You must configure the client's `sqlnet.ora`, `tnsnames.oralistener.ora` files, and configure the Microsoft Certificate Store (MCS) for authentication and encryption with X.509 certificates.

25.4.1 About Configuring TLS for Client Authentication and Encryption with X.509 Certificates

You can enable Public Key Infrastructure (PKI) authentication between Oracle Database clients and an Oracle database with X.509 certificates.

The configuration entails having to enable Public Key Infrastructure (PKI) authentication between Oracle Database clients and an Oracle database. It can be used with U.S. Federal Government Personal Identity Verification (PIV) and Department of Defense Common Access Card (CAC) cards as external keystores with the Microsoft Certificate Store (MCS) on the Windows operating system. In addition, the configuration enables Java-based Oracle Database clients to authenticate against the Oracle Database through use of client certificates stored in an Oracle wallet.

Before you begin the configuration process, note the following:

- TLS communications must run on a separate network port from normal database connections. This may affect requirements for firewall exceptions.
- TLS connections can take a longer time to establish than connections with native encryption or without any encryption, because the key exchange process introduces additional overhead.

25.4.2 Configuring the Server for Authentication and Encryption with X.509 Certificates

You must configure the server's `listener.ora`, `sqlnet.ora`, and initialization files and create a database user account for authentication and encryption with X.509 certificates.

- Step 1: Create and Configure the Server Wallet for the X.509 Certificate
  You can use the `orapki` utility to perform this configuration.
- Step 2: Shut Down the Oracle Listener on the Server
  You use different methods to shut down the Oracle listener on the server.
- Step 3: Configure the sqlnet.ora File on the Server
  You must add or modify several `sqlnet.ora` parameters on the server.
- Step 4: For Logical Volume Management, Configure the Server listener.ora File
  A logical volume management environment requires special settings for the `listener.ora` file on the server.
- Step 5: For Grid Infrastructure, Configure the Server Listener Process
  A Grid Infrastructure environment requires special settings for the `listener.ora` file on the server.
- Step 6: Set Initialization Parameters on the Server
  To avoid problems with prefixed user names, you may need to set some Oracle database initialization parameters on the server.
- Step 7: Create an External Database User on the Server
  You must create the database user by specifying the distinguished name (DN) of the user's client certificate.
- Step 8: Restart and Check the Listener Process on the Server
  If the Oracle database does not use Grid Infrastructure, then you must restart the listener on the server and check its process.

25.4.2.1 Step 1: Create and Configure the Server Wallet for the X.509 Certificate

You can use the `orapki` utility to perform this configuration.

1. Connect to the server as the `oracle` user.
2. Create a directory in which to put the server's wallet if this directory does not exist, and then `cd` to this directory.
3. Use `orapki` to create the initial wallet and give it a strong password.

   ```
   orapki wallet create -wallet wallet_file_directory -auto_login -pwd
   password
   ```

4. Generate the certificate signing request (CSR) for your server.

   Use the fully qualified domain name of the server for *`host_address`* (for example, `hostname.af.mil`). Ensure that you include the additional `O` and `C` attributes in the distinguished name as appropriate. If you do not,then the final certificate created by Federal Agency PKI will not match the request and you will not be able to import the certificate into your wallet.

   ```
   orapki wallet add -wallet wallet_file_directory -dn
   "CN=host_address,other_attributes" -asym_alg RSA -keysize 4096 -pwd
   password
   ```

5. Export the CSR so that you can submit the request to your certificate authority (CA) to generate the unique server certificate and the certificate trust chain.

   ```
   orapki wallet export -wallet wallet_file_directory -dn
   "CN=host_address,other_attributes" -request ~/host_name.csr -pwd password
   ```

If you are using Oracle Real Application Clusters (Oracle RAC), then set [HOST_ADDRESS] to the SCAN DNSname.

6. Submit the CSR (that is, *host_name*.csr) to the appropriate CA.
7. Download the appropriate root and intermediate CA certificates for your organization, any user X509 cards (CAC and PIV), and any certificates issued to non-person entities (NPEs) or service accounts.
8. Import these certificates and cards into your server wallet to establish the necessary trust chain for your server certificate and all client certificates.

   ```
   orapki wallet add -wallet wallet_file_directory -trusted_cert -cert
   cert_file_path -pwd password
   ```

   On Linux, you can import all the certificates in a single command:

   ```
   find cert_file_path -name "*.txt" -exec orapki wallet add -wallet
   wallet_file_directory -trusted_cert -cert {} -pwd password \;
   ```

9. When the signed server certificate is received, import the base64 certificate as a user certificate on the Oracle wallet on the server.

   ```
   orapki wallet add -wallet wallet_file_directory -user_cert -cert
   base64_cert_file_path -pwd password
   ```

10. As your site adds more root and intermediate CAs, update the Oracle wallet with their certificates similar to Steps 7 and 8.
11. Confirm that the server, root CA, and intermediate CA certificates are present in the Oracle wallet.

    ```
    wallet display -wallet wallet_file_directory -pwd password
    ```

    Check the Requested Certificates section of the output for a listing of the certificates.

If the Oracle database uses Grid Infrastructure, then configure the Oracle wallet directory and files located at *wallet_file_directory* to be readable by the grid user. Additionally, if it is an Oracle RAC database, then make the Oracle wallet available in a similar manner on all supporting database nodes.

25.4.2.2 Step 2: Shut Down the Oracle Listener on the Server

You use different methods to shut down the Oracle listener on the server.

Depending on your environment, use one of the following commands to stop the listener:

- If the Oracle database does not use Oracle Real Applications (Oracle RAC) or Oracle Grid Infrastructure Storage Management, then as the oracle user, use the following lsnrctl command:

  ```
  lsnrctl stop
  ```

- If the Oracle database uses Oracle Grid Infrastructure Storage Management, then as the grid user, use the following `lsnrctl` command:

  ```
  srvctl stop listener
  ```

- If the Oracle database is an Oracle RAC database, as the grid user, then use the following `srvctl` command:

  ```
  srvctl stop scan_listener
  ```

25.4.2.3 Step 3: Configure the sqlnet.ora File on the Server

You must add or modify several `sqlnet.ora` parameters on the server.

1. Back up the `sqlnet.ora` file, which is typically located in the *`ORACLE_HOME`*`/network/admin` directory.
2. Edit the `sqlnet.ora` file to include the following parameters.

   In the following settings, the `SSL_VERSION` and `SSL_CIPHER_SUITES` parameters are optional and depend on your site's requirements.

   ```
   ###Begin required parameters to be Added or Modified
   SQLNET.AUTHENTICATION_SERVICES = (beq, tcps)
   SSL_VERSION = 1.2
   SSL_CIPHER_SUITES = (TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
   TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
   TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
   TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256)
   SSL_CLIENT_AUTHENTICATION = TRUE
   WALLET_LOCATION = (SOURCE = (METHOD = FILE)(METHOD_DATA = (DIRECTORY =
   wallet_file_directory)))
   #Added when NATIVE Encryption is also configured
   SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS = TRUE
   ###End required parameters to be Added or Modified

   ###Begin optional parameters to be Added or Modified
   #SSL_CERT_REVOCATION = #set to none, requested, or required
   #SSL_CRL_PATH = #set to directory containing CRLs
   #SSL_CRL_FILE = #set to file containing CRLs
   #SSL_EXTENDED_KEY_USAGE = #set to extended key the client cert is to
   present
   ###End optional parameters
   ```

25.4.2.4 Step 4: For Logical Volume Management, Configure the Server listener.ora File

A logical volume management environment requires special settings for the `listener.ora` file on the server.

This procedure assumes that you will modify an existing `listener.ora` file. However, it also possible to configure a newly created listener by using Net Manager (`netmgr`) as well. Oracle recommends that you use a standard TCPS port setting of 2484, but you can still use another port number. Your firewalls, security lists, and network security groups must be configured to allow traffic from your clients to the TCPS port that you specify.

1. As the `oracle` user, back up the `listener.ora` file.
2. Edit the `listener.ora` file to include the following parameters:

   Ensure that you add the `ADDRESS` parameters in the order shown. Note that the `SSL_VERSION` parameter is optional and depends on your site's requirements.

```
###Modify the LISTENER parameter to add the following ADDRESS parameter
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = host_address)(PORT = 1521))
      (ADDRESS = (PROTOCOL = TCPS)(HOST = host_address)(PORT = 2484))
      (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
    )
  )

###Begin required parameters to be Added or Modified
SSL_VERSION = 1.2
SSL_CLIENT_AUTHENTICATION = TRUE

WALLET_LOCATION = (SOURCE = (METHOD = FILE)(METHOD_DATA = (DIRECTORY =
wallet_file_directory)))
###End required parameters to be Added or Modified
```

25.4.2.5 Step 5: For Grid Infrastructure, Configure the Server Listener Process

A Grid Infrastructure environment requires special settings for the `listener.ora` file on the server.

You must perform this procedure as the grid user on all nodes that are associated with the Oracle database.

1. As the grid user, back up the `listener.ora` file.
2. Edit the `listener.ora` file to include the following parameters:

   Ensure that you add the `ADDRESS` parameters in the order shown.

```
###Begin required parameters to be Added or Modified
SSL_VERSION = 1.2
SSL_CLIENT_AUTHENTICATION = TRUE
WALLET_LOCATION = (SOURCE = (METHOD = FILE)(METHOD_DATA = (DIRECTORY =
wallet_file_directory)))
###End required parameters to be Added or Modified
```

3. Add TCPS services to the listener.

```
srvctl modify listener -endpoints "TCP:1521/TCPS:2484"
```

4. If this is an Oracle Real Applications Clusters (Oracle RAC) database, then run the following command:

```
srvctl modify scan_listener -endpoints "TCP:1521/TCPS:2484"
```

25.4.2.6 Step 6: Set Initialization Parameters on the Server

To avoid problems with prefixed user names, you may need to set some Oracle database initialization parameters on the server.

1. Connect to the database as a user who has the `ALTER SYSTEM` system privilege.
2. Set the following parameters:

   ```
   ALTER SYSTEM SET OS_AUTHENT_PREFIX='' SCOPE=SPFILE;
   ```

3. Restart the database instance.

25.4.2.7 Step 7: Create an External Database User on the Server

You must create the database user by specifying the distinguished name (DN) of the user's client certificate.

Though users that are identified externally can be granted proxy privileges to connect through to other schemas (as in the case of developers accessing an application schema in a test environment), they cannot be granted privileges such as `SYSDBA` that require credentials to be stored in the database password file.

1. Connect to the database as a user who has the `CREATE USER` system privilege.
2. Create the external user as follows:

   For example, to create the external user `pfitch`:

   ```
   CREATE USER pfitch IDENTIFIED EXTERNALLY AS
   'CN=FITCH.PETER.I.1234567890,other_attributes';
   ```

3. At minimum, grant this user the `CREATE SESSION` privilege so that the user can connect to the*other_attributes* database.

   ```
   GRANT CREATE SESSION TO pfitch;
   ```

25.4.2.8 Step 8: Restart and Check the Listener Process on the Server

If the Oracle database does not use Grid Infrastructure, then you must restart the listener on the server and check its process.

Depending on your environment, use one of the following commands to restart and check the listener:

- If the Oracle database does not use Oracle Real Applications (Oracle RAC) or Oracle Grid Infrastructure Storage Management, then as the `oracle` user, use the following `lsnrctl` commands:

  ```
  lsnrctl start
  lsnrctl status
  ```

- If the Oracle database uses Oracle Grid Infrastructure Storage Management, then as the grid user, use the following `lsnrctl` commands:

```
srvctl start listener
srvctl status listener
```

- If the Oracle database is an Oracle RAC database, as the grid user, then use the following `srvctl` commands:

```
srvctl start scan_listener
srvctl status scan_listener
```

25.4.3 Configuring the Client for Authentication and Encryption with X.509 Certificates

You must configure the client's `sqlnet.ora`, `tnsnames.oralistener.ora` files, and configure the Microsoft Certificate Store (MCS) for authentication and encryption with X.509 certificates.

- Step 1: Configure the sqlnet.ora File on the Client
  You must add or modify several `sqlnet.ora` parameters on the client.
- Step 2: Configure the tnsnames.ora File on the Client
  You must modify the `tnsnames.ora` file on the client.
- Step 3: Configure Microsoft Certificate Store on the Client
  The Microsoft Certificate Store (MCS), which enables you to store and manage certificates locally, can be configured on an Oracle Database Windows client.

25.4.3.1 Step 1: Configure the sqlnet.ora File on the Client

You must add or modify several `sqlnet.ora` parameters on the client.

This configuration will enable you to use the Microsoft Certificate Store (MCS) to store and manage certificates.

1. Back up the `sqlnet.ora` file, which is typically located in the *`ORACLE_HOME`*`/network/admin` directory.
2. Edit the `sqlnet.ora` file to include the following parameters.

   The `SSL_VERSION` parameter setting depends on your site's requirements.

```
###Begin required parameters to be Added or Modified
SQLNET.AUTHENTICATION_SERVICES = (nts, tcps)

SSL_VERSION = 1.2

SSL_SERVER_DN_MATCH = TRUE

WALLET_LOCATION = (SOURCE = (METHOD = MCS))

###Begin optional parameters to be Added or Modified
#SSL_CIPHER_SUITES = algorithms to be used for TLS encryption
###End optional parameters
```

25.4.3.2 Step 2: Configure the tnsnames.ora File on the Client

You must modify the `tnsnames.ora` file on the client.

1. Back up the `tnanames.ora` file, which is typically located in the *ORACLE_HOME*`/network/admin` directory.
2. Edit the `tnsnames.ora` file to include the following parameters:

```
service_alias =
    (DESCRIPTION =
            (ADDRESS =
                (PROTOCOL = TCPS)
                (HOST = host_ip_address)
                (PORT = 2484)
            )
            (CONNECT_DATA =
                (SERVICE_NAME = database_service_name)
            )
            (SECURITY =
                (SSL_SERVER_CERT_DN = "CN=host_ip_address,other_attributes)
            )
    )
```

25.4.3.3 Step 3: Configure Microsoft Certificate Store on the Client

The Microsoft Certificate Store (MCS), which enables you to store and manage certificates locally, can be configured on an Oracle Database Windows client.

- About Configuring Microsoft Certificate Store on the Client
  Before you configure Microsoft Certificate Store (MCS) on the client, you should ensure that your client environment is properly set up.
- Setting the TNS_ADMIN Environment Variable
  The `TNS_ADMIN` environment variable must be set in a special way to facilitate the MCS operation.
- Configuring Microsoft Certificate Store on the Client
  For the mTLS configuration to work, the certificates for the root and intermediate CAs that signed the certificate that the database server used must be added to the MCS.
- Testing the Microsoft Certificate Store Configuration Using tnsping
  The `tnsping` utility determines whether an Oracle service can be successfully reached.
- Testing the Microsoft Certificate Store Configuration Using SQL*Plus
  SQL*Plus is the most basic Oracle Database utility commonly used by users, administrators, and programmers that can be used to confirm mTLS and user authentication to the database.

25.4.3.3.1 About Configuring Microsoft Certificate Store on the Client

Before you configure Microsoft Certificate Store (MCS) on the client, you should ensure that your client environment is properly set up.

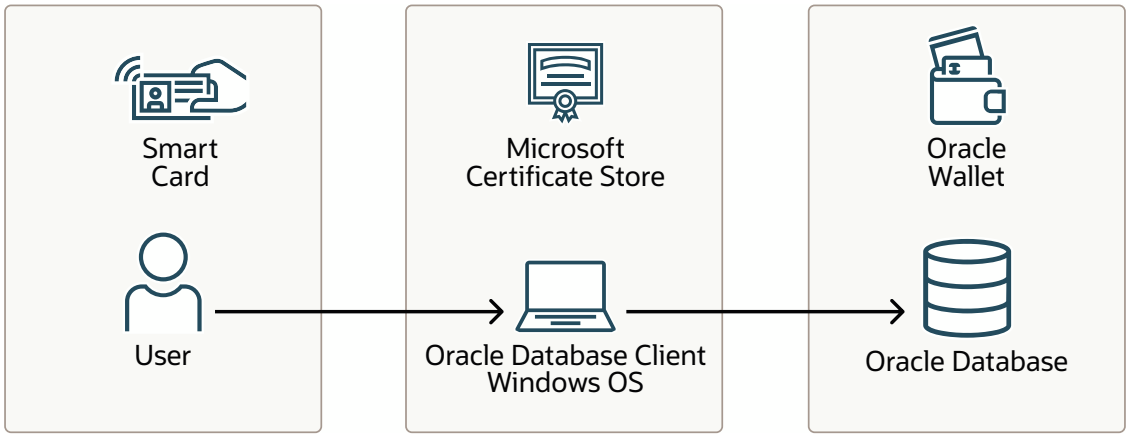
These instructions assume the following:

- The Oracle Database client has been installed and configured to communicate with the Oracle Database server.
- All clients have the latest patches installed.
- You have installed the appropriate hardware and software to enable MCS to read the certificates from the X509 smart cards (Common Access Card (CAC), Personal Identity Verification (PIV)

You can also configure MCS to work on the client with SQL Developer and with Java using JDBC Type 4 Drivers. See My Oracle Support note 2959952.1.

The following diagram illustrates a smart card and MCS in an Oracle Database environment.

Figure 25-1 Smart Card and MCS in an Oracle Database Environment



In this diagram:

1. A user logs in to the Oracle database. The user's user certificate, private key, and other necessary certificates are on the smart card.
2. The database connection from the client is configured to use MCS.
3. The wallet in the Oracle database is a PKCS11 wallet with a private key an certificate. The Oracle Database wallet holds the server private key and the trusted root certificate.

25.4.3.3.2 Setting the TNS_ADMIN Environment Variable

The `TNS_ADMIN` environment variable must be set in a special way to facilitate the MCS operation.

The following setting enables a user to place all necessary `*.ora` files within their own user profile where they have ownership and control. It also allows each user of a system to have individual, personalized configurations.

1. Open the System Properties window on Windows. (Search for Advanced System Settings.)
2. Select the **Advanced** tab.
3. Click **Environment Variables**.
4. In the Environment Variables window, if `TNS_ADMIN` is not listed, then click **New**. If it is listed, then click **Edit**.

5. In the New (or Edit) User Variable dialog box, enter the following value in the **Variable value** field:

   ```
   %USERPROFILE%\Oracle\admin
   ```

6. Click **OK**.

25.4.3.3.3 Configuring Microsoft Certificate Store on the Client

For the mTLS configuration to work, the certificates for the root and intermediate CAs that signed the certificate that the database server used must be added to the MCS.

1. Download the certificates for the root and intermediate CAs that were used to sign the database server certificate when you created and configured the server wallet.
2. Start the MCS Certificate Import wizard.
3. In the Welcome to the Certificate Import Wizard page, select the **Current User** option, and then click **Next**.
4. On the Certificate Store page, select the **Automatically select the certificate store based on the type of certificate** option, and then click **Next**.
5. In the Completing the Certificate Import Wizard page, check the settings that you made, and then click **Finish**. Click **OK** in the Certificate Import Wizard confirmation window.
6. Confirm that the CAs have successfully been loaded into MCS.
   a. In the Console Root tree on the left, under Certificates - Current User, expand the Trusted Root Certificates folder.
   b. Select the Certificates folder to display the Certificate window.
   c. Check the contents. The window will describe the purpose of the certificate, who it was issued to, who issued it, and the dates the certificate will be valid for. Click **OK** to dismiss the window.

Related Topics

- Step 1: Create and Configure the Server Wallet for the X.509 Certificate
  You can use the `orapki` utility to perform this configuration.

25.4.3.3.4 Testing the Microsoft Certificate Store Configuration Using tnsping

The `tnsping` utility determines whether an Oracle service can be successfully reached.

1. On the client, confirm that there is TCP/IP connectivity to the TLS port (that is, 2484) configured from the client to the database using your utility of choice.

   If there does not appear to be connectivity, work with your network and system administrators to confirm that the appropriate firewall, security list, network security groups, and so on are a configured to allow the communication.

2. Run the `tnsping` command (by default in the *`ORACLE_HOME`*`/bin` directory) against the service alias that you defined in the `tnsnames.ora` file.

   ```
   tnsping service_alias
   ```

3. When prompted, select the certificate that you associated with the external Oracle Database user account that you created earlier.

After you provide the Personal Identification Number (PIN) for the certificate, output similar to the following appears:

```
Used parameter files:

[ORACLE_HOME]\network\admin\sqlnet.ora

Used TNSNAMES adapter to resolve the alias

Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS) (HOST =
host_address) (PORT = 2484)) (CONNECT_DATA = (SERVICE_NAME =
database_service_name]))
 (SECURITY = (SSL_SERVER_CERT_DN = CN=host_addres,other_attributes)))

OK (4920 msec)
```

The response time may seem large. The elapsed time shown includes the amount of time it takes the user to react to the prompt and select a certificate, so it will always be several seconds.

Related Topics

- Step 2: Configure the tnsnames.ora File on the Client
  You must modify the `tnsnames.ora` file on the client.

25.4.3.3.5 Testing the Microsoft Certificate Store Configuration Using SQL*Plus

SQL*Plus is the most basic Oracle Database utility commonly used by users, administrators, and programmers that can be used to confirm mTLS and user authentication to the database.

1. On the client, run SQL*Plus against the service alias you defined earlier in the client `tnsnames.ora` file.

   ```
   sqlplus /@service_alias
   ```

2. When prompted, select the certificate that you associated with the external Oracle Database user account that you created earlier.

   After you provide the Personal Identification Number (PIN) for the certificate, output similar to the following appears:

   ```
   SQL*Plus: Release release - Production on Mon May 23 14:03:10 2022

   Version release

   Copyright (c) 1982, 2019, 2023 Oracle.  All rights reserved.

   Last Successful login time: Wed Oct 18 2023 16:47:43 +00:00

   Connected to:

   Oracle Database release - Production

   Version release
   ```

3. Confirm that you are connected as the user associated with the client certificate you used.

```
show user;
```

4. Confirm that the TCPS protocol is being used.

```
SELECT SYS_CONTEXT('USERENV','NETWORK_PROTOCOL') FROM DUAL;
```

Output similar to the following should appear:

```
SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
------------------------------------------------
tcps
```

Related Topics

- Step 2: Configure the tnsnames.ora File on the Client
  You must modify the `tnsnames.ora` file on the client.

25.5 Configuring Email over Transport Layer Security with an Oracle Wallet

You can use an Oracle wallet, PL/SQL packages, and security access control lists (ACLs) to configure email over a Transport Layer Security (TLS) connection.

1. Use `openssl` to get the URL certificates from the mail server.

   You can perform this step with email server, to dump the certificate chain to a standard output (`stdout`). Typically, this command dumps the server certificate (`cert 0`) and the intermediate trusted certificate (`cert 1...n`). For example:

```
$ openssl s_client -showcerts -connect office365.com:443
```

   Output similar to the following appears:

```
depth=2 C = US, O = DigiCert Inc, CN = DigiCert Global Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert Cloud Services CA-1
verify return:1
depth=0 C = US, ST = Washington, L = Redmond, O = Microsoft Corporation,
CN = outlook.com
verify return:1
---
Certificate chain
0 s:/C=US/ST=Washington/L=Redmond/O=Microsoft Corporation/CN=outlook.com
i:/C=US/O=DigiCert Inc/CN=DigiCert Cloud Services CA-1
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
...
DONE
```

2. Copy and paste the certificates in this output to text files with the extension `.cer`.

You must copy the text that appears after `-----BEGIN CERTIFICATE -----` and before `-----END CERTIFICATE-----`. Example files are as follows:

- `file_root.cer`
- `file_trusted.cer`
- `file_user.cer`

3. Check the CA issuer and the CA subject of each certificate that you copied to a certificate file.

   The CA issuer is the company that created the certificate and the subject indicates the information that had been provided when the certificate was created.

   - To check the root certificate:

     ```
     openssl x509 -in file_root.cer -text | grep -i issuer
     Issuer: C=US, O=DigiCert Inc, CN=DigiCert Global Root CA

     openssl x509 -in file_root.cer -text | grep -i subject
     Subject: C=US, O=DigiCert Inc, CN=DigiCert Global Root CA
     ```

   - To check the trusted certificate:

     ```
     openssl x509 -in file_trusted.cer -text | grep -i issuer
     Issuer: C=US, O=DigiCert Inc, CN=DigiCert Global Root CA

     openssl x509 -in file_trusted.cer -text | grep -i subject
     Subject: C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
     ```

   - To check the user certificate:

     ```
     openssl x509 -in file_user.cer -text | grep -i issuer
     Issuer: C=US, O=DigiCert Inc, CN=DigiCert Global Root CA

     openssl x509 -in file_user.cer -text | grep -i subject
     Subject: C=US, O=DigiCert Inc, CN=DigiCert SHA2 Secure Server CA
     ```

4. Create a folder location.

   For example:

   ```
   mkdir app/oracle/product/network/admin/email
   ```

5. Create the wallet and add its certificates to this wallet.

   a. Create an empty wallet.

      For example:

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

      If you omit the `pwd` prompt, then a password prompt appears. For better security, enter the password at the prompt instead of entering it at the command line.

b. Put the certificate into the wallet. For example:

```
orapki wallet add -wallet wallet_file_directory -trusted_cert -cert
trusted.cer
[-pwd wallet_password]
```

6. Prepare the email SQL code.

For example:

```
################################################################################
###
##
DECLARE
k_host CONSTANT VARCHAR2(100) := 'us.example.com';
k_port CONSTANT INTEGER := 587;
k_wallet_path CONSTANT VARCHAR2(100) :=
'file:app/oracle/product/network/admin/email';
k_wallet_password CONSTANT VARCHAR2(100) := 'wallet_password';
k_domain CONSTANT VARCHAR2(100) := 'localhost';
k_username CONSTANT VARCHAR2(100) := 'email_account';
k_password CONSTANT VARCHAR2(100) := 'email_account_password';
k_sender CONSTANT VARCHAR2(100) := 'email_account';
k_recipient CONSTANT VARCHAR2(100) := 'email_account_sending_too';
k_subject CONSTANT VARCHAR2(100) := 'Test TLS mail';
k_body CONSTANT VARCHAR2(100) := 'We Love Database Security';

l_conn utl_smtp.connection;
l_reply utl_smtp.reply;
l_replies utl_smtp.replies;
BEGIN
dbms_output.put_line('utl_smtp.open_connection');

l_reply := utl_smtp.open_connection
( host => k_host
, port => k_port
, c => l_conn
, wallet_path => k_wallet_path
, wallet_password => k_wallet_password
, secure_connection_before_smtp => FALSE
);

IF l_reply.code != 220
THEN
raise_application_error(-20000, 'utl_smtp.open_connection: '||
l_reply.code||'
- '||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.ehlo');

l_replies := utl_smtp.ehlo(l_conn, k_domain);

FOR ri IN 1..l_replies.COUNT
LOOP
dbms_output.put_line(l_replies(ri).code||' - '||l_replies(ri).text);
```

```
END LOOP;

dbms_output.put_line('utl_smtp.starttls');

l_reply := utl_smtp.starttls(l_conn);

IF l_reply.code != 220
THEN
raise_application_error(-20000, 'utl_smtp.starttls: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.ehlo');

l_replies := utl_smtp.ehlo(l_conn, k_domain);

FOR ri IN 1..l_replies.COUNT
LOOP
dbms_output.put_line(l_replies(ri).code||' - '||l_replies(ri).text);
END LOOP;

dbms_output.put_line('utl_smtp.auth');

l_reply := utl_smtp.auth(l_conn, k_username, k_password,
utl_smtp.all_schemes);

IF l_reply.code != 235
THEN
raise_application_error(-20000, 'utl_smtp.auth: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.mail');

l_reply := utl_smtp.mail(l_conn, k_sender);

IF l_reply.code != 250
THEN
raise_application_error(-20000, 'utl_smtp.mail: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.rcpt');

l_reply := utl_smtp.rcpt(l_conn, k_recipient);

IF l_reply.code NOT IN (250, 251)
THEN
raise_application_error(-20000, 'utl_smtp.rcpt: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.open_data');

l_reply := utl_smtp.open_data(l_conn);
```

```
IF l_reply.code != 354
THEN
raise_application_error(-20000, 'utl_smtp.open_data: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.write_data');

utl_smtp.write_data(l_conn, 'From: '||k_sender||utl_tcp.crlf);
utl_smtp.write_data(l_conn, 'To: '||k_recipient||utl_tcp.crlf);
utl_smtp.write_data(l_conn, 'Subject: '||k_subject||utl_tcp.crlf);
utl_smtp.write_data(l_conn, utl_tcp.crlf||k_body);

dbms_output.put_line('utl_smtp.close_data');

l_reply := utl_smtp.close_data(l_conn);

IF l_reply.code != 250
THEN
raise_application_error(-20000, 'utl_smtp.close_data: '||l_reply.code||' -
'||l_reply.text);
END IF;

dbms_output.put_line('utl_smtp.quit');

l_reply := utl_smtp.quit(l_conn);

IF l_reply.code != 221
THEN
raise_application_error(-20000, 'utl_smtp.quit: '||l_reply.code||' -
'||l_reply.text);
END IF;

EXCEPTION
WHEN utl_smtp.transient_error
OR utl_smtp.permanent_error
THEN
BEGIN
utl_smtp.quit(l_conn);
EXCEPTION
WHEN utl_smtp.transient_error
OR utl_smtp.permanent_error
THEN
NULL;
END;

raise_application_error(-20000, 'Failed to send mail due to the following
error: '||SQLERRM);

END;
/
```

Ensure that you set the `secure_connection_before_smtp` parameter to `FALSE`. This translates to "do not use TLS before the email is sent". Setting it to `TRUE` generates the following error if we only want to send the email over TLS:

```
ERROR at line 1:
ORA-29019: The protocol version is incorrect.
ORA-06512: at "SYS.UTL_TCP", line 63
ORA-06512: at "SYS.UTL_TCP", line 314
ORA-06512: at "SYS.UTL_SMTP", line 177
ORA-06512: at line 20
```

7. Create the user who will send emails.

   For example:

   ```
   CREATE USER user_name IDENTIFIED BY password;
   GRANT CREATE SESSION TO user_name;
   ```

8. Append the host and wallet access control entries (ACE) to the default access control list (ACL).

   a. Append the host access control entry (ACE).

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
   host => 'us.example.com',
   lower_port => 587,
   upper_port => 587,
   ace => xs$ace_type(privilege_list => xs$name_list('http'),
   principal_name => 'user_name',
   principal_type => xs_acl.ptype_db));
   END;
   /

   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
   host => 'us.example.com',
   lower_port => 587,
   upper_port => 587,
   ace => xs$ace_type(privilege_list => xs$name_list('connect'),
   principal_name => 'user_name',
   principal_type => xs_acl.ptype_db));
   END;
   /

   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
   host => 'us.example.com',
   lower_port => null,
   upper_port => null,
   ace => xs$ace_type(privilege_list => xs$name_list('resolve'),
   principal_name => 'user_name',
   principal_type => xs_acl.ptype_db));
   END;
   /
   ```

b. Append the wallet ACE.

```
BEGIN
DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE(
wallet_path =>
'file:/u01/64bit/app/oracle/product/network/admin/email',
ace => xs$ace_type(privilege_list =>
xs$name_list('use_client_certificates',
'use_passwords'),
principal_name => 'user_name',
principal_type => xs_acl.ptype_db));
END;
/
```

25.6 Troubleshooting Transport Layer Security Errors

Oracle provides a utility to help troubleshoot PKI certificate configurations as well as additional guidance below. A utility is available through the support website to review and provide feedback on your PKI certificate authentication client and server configuration.

See DBSecChk Utility 2.0.0.5 (Doc ID 3066006.1).

- Step 1: Check the TLS Connection with the tnsping Utility
  A successful connection using the `tnsping` utility shows that the database service has been registered to the listener on the TCPS endpoint.
- Step 2: Check the SSL_VERSION Parameter
  An incorrectly set `SSL_VERSION` parameter can cause Transport Layer Security (TLS) problems.
- Step 3: Check the Wallet File Permissions
  The Transport Layer Security (TLS) connection requires the database and listener to have access to the auto-login wallet file (`cwallet.sso`).
- Step 4: Check the Wallet Settings in the sqlnet.ora and listener.ora Files
  Transparent Layer Security (TLS) problems can arise from wallet and certificate configuration errors in the `sqlnet.ora` and `listener.ora` files.
- Step 5: Enable Tracing for the SQL*Net and Listener Connections
  In the `sqlnet.ora` file, you can enable tracing for SQL*Net and listener connections.

25.6.1 Step 1: Check the TLS Connection with the tnsping Utility

A successful connection using the `tnsping` utility shows that the database service has been registered to the listener on the TCPS endpoint.

- On the server on which the Oracle database is installed, run the `tnsping` command at the command line using the following syntax:

  `tnsping net_service_name [count]`

  For example:

  ```
  tnsping sales count
  ```

In this specification:

- *net_service_name* (sales) is the service name that is specified in the tnsnames.ora file, or it can be the name service that is in use, such as NIS.
- count, which is optional, determines how many times the program attempts to reach the server.

Output similar to the following appears:

```
TNS Ping Utility for Linux: Version 23.0.0.0.0 - Production on 26-APR-2023
18:21:47

Copyright (c) 1997, 2023, Oracle. All rights reserved.

Used parameter files:
$ORACLE_HOME/network/admin/sqlnet.ora

Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCPS)(HOST =
host_name)(PORT = port)) (CONNECT_DATA = (SERVER = DEDICATED)
(SERVICE_NAME = sales)))
OK (30 msec)
```

If the test fails with an TNS-12560: NS:protocol adapter error error, then ensure that the lines in the sqlnet.ora and listener.ora files do not have leading spaces. If the connection still has errors, then you must investigate further, such as checking the permissions of wallet files or other settings.

See *Oracle Database Net Services Administrator's Guide* for detailed information about using the tnsping utility.

25.6.2 Step 2: Check the SSL_VERSION Parameter

An incorrectly set SSL_VERSION parameter can cause Transport Layer Security (TLS) problems.

You should ensure that the SSL_VERSION parameter in the server and client sqlnet.ora file is set to the correct version of TLS, so that connections can be established. For example:

```
SSL_VERSION= TLSv1.3
```

By default, Oracle Database uses the most secure protocol that is available when SSL_VERSION is not set.

See *Oracle Database Net Services Reference* to learn more about how to set the SSL_VERSION parameter for the correct version of TLS.

25.6.3 Step 3: Check the Wallet File Permissions

The Transport Layer Security (TLS) connection requires the database and listener to have access to the auto-login wallet file (cwallet.sso).

In the case of an Oracle Real Application Clusters (Oracle RAC) database, both the Grid Infrastructure Oracle Home owner and the Database Oracle Home owner must have access to the contents of a cwallet.sso file containing the correct certificates. Quite often the configuration implies the usage of the same cwallet.sso file for both environments, in which

case the permissions should be set appropriately so that both users can have access to the file no matter who is the owner of the file.

By default, the wallet permissions are as follows:

```
$ ls -ltr

-rw-------. 1  ewallet.p12
-rw-------. 1  cwallet.sso
```

If the `cwallet.sso` file will be used by the Grid Infrastructure Oracle Home owner (usually `grid`) then user `grid` must be a member of the `oinstall` group. You can change the permissions as follows:

```
$ chmod 640 cwallet.sso
$ ls -ltr

-rw-------. 1 oracle oinstall 75 Mar 6 10:47 ewallet.p12
-rw-r-----. 1 oracle oinstall 120 Mar 6 10:47 cwallet.sso
```

25.6.4 Step 4: Check the Wallet Settings in the sqlnet.ora and listener.ora Files

Transparent Layer Security (TLS) problems can arise from wallet and certificate configuration errors in the `sqlnet.ora` and `listener.ora` files.

These settings enable you to encrypt the connections between the database and its clients. (Another way to handle this encryption is with the external network services PL/SQL packages, `UTL_SMTP`, `UTL_HTTP`, and `UTL_TCP`.)

Note the following:

- **For the server:** Set the `WALLET_ROOT` parameter. (The `WALLET_LOCATION` parameter can still be used.) Both trusted certificate and server certificate are required.
- **For the client:** Set the `WALLET_LOCATION` in `sqlnet.ora`. Only trusted certificates are required if one-way TLS is configured. If mTLS is configured, then both trusted certificate and server certificate are required.
- **For the listener:** Set the `WALLET_LOCATION` parameter in the `listener.ora` file. Both trusted certificate and server certificate are required.

An example `WALLET_LOCATION` parameter setting is as follows:

```
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
      (METHOD_DATA =
        (DIRECTORY = wallet_location)
      )
 )
```

The certificates can be self-signed or they can be signed by a third-party authority.

You can use the `orapki wallet display -wallet` command to view the contents of a wallet to find if it has self-signed certificates. For example:

```
$ orapki wallet display -wallet .

Requested Certificates:
User Certificates:
Subject: C=US,CN=MYROOT
Trusted Certificates:
Subject: C=US,CN=MYROOT
```

The following example shows the output for a wallet that has wallet that has certificates that were provided by a third-party authority:

```
Requested Certificates:
User Certificates:
Subject: CN=*.us.example.com,O=Example Corporation,L=Redwood
City,ST=California,C=US
Trusted Certificates:
Subject: CN=DigiCert Global Root CA,O=DigiCert Inc,C=US
Subject: CN=DigiCert TLS RSA SHA256 2020 CA1,O=DigiCert Inc,C=US
```

25.6.5 Step 5: Enable Tracing for the SQL*Net and Listener Connections

In the `sqlnet.ora` file, you can enable tracing for SQL*Net and listener connections.

For example, to enabling tracing for SQL*Net:

```
TRACE_LEVEL_CLIENT=SUPPORT
TRACE_DIRECTORY_CLIENT=trace_dir
TRACE_LEVEL_SERVER=SUPPORT
TRACE_DIRECTORY_SERVER=trace_dir
DIAG_ADR_ENABLED=OFF
```

For the listener, you can set the following tracing parameters:

```
TRACE_FILE_LISTENER = LISTENER.TRC
TRACE_DIRECTORY_LISTENER = trace_dir
TRACE_LEVEL_LISTENER = SUPPORT
TRACE_FILELEN_LISTENER = 10240
TRACE_FILENO_LISTENER=10
```

The following output indicates that the TLS connection failed because the wrong TLS protocol was used. To find how to address these errors, see My Oracle Support note 244527.1.

```
[<DATE AND TIME>] ntzdosecneg: entry
[<DATE AND TIME>] nttrd: entry
[<DATE AND TIME>] nttrd: socket 13 had bytes read=11
[<DATE AND TIME>] nttrd: exit
[<DATE AND TIME>] ntzdosecneg: SSL handshake failed with error 29019.
[<DATE AND TIME>] ntzdosecneg: exit
[<DATE AND TIME>] ntzcontrol: failed with error 542
[<DATE AND TIME>] ntzcontrol: exit
```

```
[<DATE AND TIME>] nserror: entry
[<DATE AND TIME>] nserror: nsres: id=0, op=79, ns=12561, ns2=0; nt[0]=0,
nt[1]=0, nt[2]=0; ora[0]=0, ora[1]=0, ora[2]=0
[<DATE AND TIME>] nsclose: entry
[<DATE AND TIME>] nsvntx_dei: entry
[<DATE AND TIME>] nsvntx_dei: exit
```

See Troubleshooting the Transport Layer Security Configuration for information about common error codes.

See also *Oracle Database Net Services Administrator's Guide* for more information about using trace settings to track connections.