Oracle® Database

Database Net Services Reference

23ai
F46840-04
May 2024

Oracle Database Database Net Services Reference, 23ai

F46840-04

Primary Author: Binika Kumar

Contributing Authors: Doug Williams, Prakash Jashnani

Contributors: Alan Williams, Abhishek Dadhich, Anita Patel, Ashish Trivedi, Bhaskar Gharu, Bhaskar Mathur, Ching Tai, Christopher Jones, David Lin, Deepak Yadav, Feroz Khan, Gulshan Kumar, Hector Pujol, Jean Zeng, Kant Patel, Kevin Neel, Krishna Itikarlapalli, Kunal Waghmare, Mark Dilman, Michael McMahon, Misaki Miyashita, Manjunatha Kuruba, Mohammad Raihan Afzal, Murali Purayathu, Norman Woo, Peter Knaggs, Rajbir Chahal, Robert Achacoso, Santanu Datta, Saravanakumar Ramasubramanian, Sarma Namuduri, Scot McKinley, Seshan Parameswaran, Srinivas Pamu, Steve Ding, Sudarshan Soma, Sudeep Reguna, Sweta Mogra, Thanigai Nallathambi, Vikram Kumar, Yi Ouyang

Contents

2 Oracle Connection Manager Control Utility

6 Local Naming Parameters in the tnsnames.ora File

9 Oracle Connection Manager Parameters

Preface

Review this publication to obtain a complete listing and description of control utility commands and configuration file parameters that you can use to manage Oracle Net Services components.

Audience

Oracle Database Net Services Reference is intended for network administrators who are responsible for configuring and administering network components.

To use this document, you should be familiar with the networking concepts and configuration tasks described in *Oracle Database Net Services Administrator's Guide*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Listener Control Utility

This chapter describes the Listener Control Utility commands and syntax. The terms **SQL*Net** and **Net Services** are used interchangeably throughout Oracle documentation.

- Listener Control Utility Overview
  To perform basic management functions on one or more listeners, you can use the Listener Control utility commands. You can also use these commands to view and change parameter settings.
- SET and SHOW Listener Control Utility Commands
  The `SET` and `SHOW` commands enable you to alter and view listener configuration parameters.
- Understanding Distributed Operations
  Listener Control utility can perform operations on local or remote listeners.
- Understanding Oracle Net Listener Security
  Authentication for listener administration depends on whether you access the listener locally or remotely.
- Listener Control Utility Commands
  Use Listener Control Utility commands to manage and configure your listeners.

1.1 Listener Control Utility Overview

To perform basic management functions on one or more listeners, you can use the Listener Control utility commands. You can also use these commands to view and change parameter settings.

The Listener Control utility enables you to administer listeners. The syntax of Listener Control utility commands is as follows:

```
lsnrctl command listener_name
```

In the preceding command, *`listener_name`* is the name of the listener that you want to administer. If you do not specify a specific listener in the command string, then the command is directed to the default listener name, `LISTENER`.

You can also issue Listener Control utility commands at the `LSNRCTL>` program prompt. To obtain the prompt, enter `lsnrctl` with no arguments at the operating system command line. When you run `lsnrctl`, the program is started. You can then enter the commands from the program prompt. The basic syntax of issuing commands from `LSNRCTL>` program prompt is:

```
lsnrctl
LSNRCTL> command listener_name
```

You can combine commands in a standard text file and then run them as a sequence of commands. To run in batch mode, use this format:

```
lsnrctl @file_name
```

To identify comments in the batch script, you can use either `REM` or `#`. All other lines are considered commands. Commands that require confirmation do not require confirmation during batch processing.

For most commands, Listener Control utility establishes an Oracle Net connection with the listener that transmits the command. To initiate an Oracle Net connection to the listener, Listener Control utility must obtain the protocol addresses for the named listener or a listener named `LISTENER`. This is done by resolving the listener name with one of the following:

- `listener.ora` file in the directory specified by the `TNS_ADMIN` environment variable.
- `listener.ora` file in the *`ORACLE_HOME`*`/network/admin` directory.
- Naming method; for example, a `tnsnames.ora` file.

If none of the preceding mechanisms resolve the listener name, then Listener Control utility uses the default listener name `LISTENER`, resolves the host name IP address, and uses port 1521.

The Listener Control utility supports the following types of commands:

- Operational commands, such as `START`, and `STOP`.
- Modifier commands, such as `SET TRC_LEVEL`.
- Informational commands, such as `STATUS`, and `SHOW LOG_FILE`.

1.2 SET and SHOW Listener Control Utility Commands

The `SET` and `SHOW` commands enable you to alter and view listener configuration parameters.

Use the `SET` command to alter parameter values for a specified listener. You set the name of the listener to administer using the `SET CURRENT_LISTENER` command. Parameter values remain in effect until the listener is shut down. If you want these settings to persist, then use the `SAVE_CONFIG` command to save changes to the `listener.ora` file.

You can use the `SHOW` command to display the current value of a configuration setting.

1.3 Understanding Distributed Operations

Listener Control utility can perform operations on local or remote listeners.

This example explains how to configure listeners for remote administration.

Set Up a Computer to Remotely Administer a Listener

Ensure that the Listener Control utility (`lsnrctl`) executable is installed in the *`ORACLE_HOME`*`/bin` directory. You can resolve the name of the listener that you want to administer either through a `listener.ora` file, or by a naming method.

When you administer a listener remotely, you can issue all of the commands except `START`. However, Listener Control utility only starts the listener on the same computer from which the utility runs.

When issuing commands, specify the listener name as an argument. If you omit the listener name in the command, then the listener name set with the command `SET CURRENT_LISTENER` is used. If you do not set the listener name with that command, then the command is directed to the default listener name, `LISTENER`.

Example 1-1 Issuing Commands Using the Listener Control Utility

```
LSNRCTL> SERVICES lsnr
```

1.4 Understanding Oracle Net Listener Security

Authentication for listener administration depends on whether you access the listener locally or remotely.

Local listener administration security is administered using local operating system authentication. Local authentication restricts listener administration to the user account that started the listener or to the super user. By default, remote listener administration is disabled.

Oracle recommends that you perform listener administration in the default mode and that you access the system remotely using a remote login. When you administer the listener remotely, use either Oracle Enterprise Manager Cloud Control or Secure Shell (SSH) to access the remote host.

1.5 Listener Control Utility Commands

Use Listener Control Utility commands to manage and configure your listeners.

- EXIT
  Use the Listener Control utility command `EXIT` to exit from the Listener Control utility and return to the operating system prompt.
- HELP
  Use the Listener Control utility command `HELP` to list the Listener Control utility commands and to obtain syntax help for particular Listener Control utility commands.
- QUIT
  Use the Listener Control utility command `QUIT` to exit from the Listener Control utility and return to the operating system prompt.
- RELOAD
  Use the Listener Control utility command `RELOAD` to reload the `listener.ora` file so that you can add or change statically configured services without stopping the listener.
- SAVE_CONFIG
  Use the Listener Control utility command `SAVE_CONFIG` to save the current configuration state of the listener to the `listener.ora` file.
- SERVICES
  Use the Listener Control utility command `SERVICES` to return details about database services, instances, and service handlers to which listeners forward client connection requests.
- SET
  Use the Listener Control utility command `SET` to alter listener parameter values.

- SET CURRENT_LISTENER
  Use the Listener Control utility command `SET CURRENT_LISTENER` to set the name of the listener to administer.
- SET DISPLAYMODE
  Use the Listener Control utility command `SET DISPLAYMODE` to change the format and detail level for the `SERVICES` and `STATUS` commands.
- SET INBOUND_CONNECT_TIMEOUT
  Use the Listener Control utility command `SET INBOUND_CONNECT_TIMEOUT` to specify the duration in which clients must complete connection requests to listeners after establishing network connections.
- SET LOG_DIRECTORY
  Use the Listener Control utility command `SET LOG_DIRECTORY` to set the destination directory to which the listener log file is written.
- SET LOG_FILE
  Use the Listener Control utility command `SET LOG_FILE` to set the listener log file name.
- SET LOG_STATUS
  Use the Listener Control utility command `SET LOG_STATUS` to turn listener logging on or off.
- SET SAVE_CONFIG_ON_STOP
  Use the Listener Control utility command `SET SAVE_CONFIG_ON_STOP` to specify whether listener parameter value changes that you make with the `SET` command are saved to the `listener.ora` file when you stop the listener with the `STOP` command.
- SET TRC_DIRECTORY
  Use the Listener Control utility command `SET TRC_DIRECTORY` to set the destination directory into which Oracle writes listener trace files.
- SET TRC_FILE
  Use the Listener Control utility command `SET TRC_FILE` to set the names of listener trace files.
- SET TRC_LEVEL
  Use the Listener Control utility command `SET TRC_LEVEL` to set a specific listener tracing level.
- SHOW
  Use the Listener Control utility command `SHOW` to list current listener parameter values.
- SHOW STATS
  Use the Listener Control utility `SHOW STATS` command with the `-REG` option to display statistics about database service registration commands.
- SPAWN
  Use the Listener Control utility command `SPAWN` to start a program that is stored on the computer on which the listener is running and that is listed with an alias in the `listener.ora` file.
- START
  Use the Listener Control utility command `START` to start a named listener.

- STATUS
  Use the Listener Control utility command `STATUS` to show listener status information.
- STOP
  Use the Listener Control utility command `STOP` to stop the named listener.
- TRACE
  Use the Listener Control utility command `TRACE` to set listener tracing.
- VERSION
  Use the Listener Control utility command `VERSION` to show the current version of Listener Control utility.

1.5.1 EXIT

Use the Listener Control utility command `EXIT` to exit from the Listener Control utility and return to the operating system prompt.

Purpose

To exit from the Listener Control utility and return to the operating system prompt.

Prerequisites

None

Syntax

From the Listener Control utility:

```
LSNRCTL> EXIT
```

Arguments

None

Usage Notes

This command is identical to the `QUIT` command.

Example

```
LSNRCTL> EXIT
```

1.5.2 HELP

Use the Listener Control utility command `HELP` to list the Listener Control utility commands and to obtain syntax help for particular Listener Control utility commands.

Purpose

To provide a list of the Listener Control utility commands or to provide syntax help for a particular Listener Control utility command.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl HELP command
```

From the Listener Control utility:

```
LSNRCTL> HELP command
```

Arguments

command: The Listener Control utility command. Commands are shown in the following example output.

When you enter a command as an argument to `HELP`, the Listener Control utility displays information about how to use the command. When you enter `HELP` without an argument, the Listener Control utility lists all the commands.

Example

```
LSNRCTL> HELP
The following operations are available
An asterisk (*) denotes a modifier or extended command:
exit
quit
reload
services
set*
show*
spawn
start
status
stop
trace
version
```

1.5.3 QUIT

Use the Listener Control utility command `QUIT` to exit from the Listener Control utility and return to the operating system prompt.

Purpose

To exit from the Listener Control utility and return to the operating system prompt.

Prerequisites

None

Syntax

From the Listener Control utility:

```
LSNRCTL> QUIT
```

Arguments

None

Usage Notes

This command is identical to the `EXIT` command.

Example

```
LSNRCTL> QUIT
```

1.5.4 RELOAD

Use the Listener Control utility command `RELOAD` to reload the `listener.ora` file so that you can add or change statically configured services without stopping the listener.

Purpose

To reload the `listener.ora` file. This command enables you to add or change statically configured services without actually stopping the listener.

When you run this command, the database services, instances, service handlers, and listening endpoints previously registered dynamically with the listener are unregistered, and subsequently registered again.

To obtain a lightweight reload without dropping registration, use the option `-with_ha`. Using this option ensures that registered services remain available to clients during reload.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl RELOAD [-with_ha] listener_name
```

From the Listener Control utility:

```
LSNRCTL> RELOAD [-with_ha] listener_name
```

Arguments

`listener_name`: The listener name, if the default name of `LISTENER` is not used.

`-with_ha`: command option used with `RELOAD` that indicates that the reload of `listener.ora` is completed without dropping existing registrations.

Example

```
LSNRCTL> RELOAD
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)
(PORT=1521)))
The command completed successfully
```

1.5.5 SAVE_CONFIG

Use the Listener Control utility command `SAVE_CONFIG` to save the current configuration state of the listener to the `listener.ora` file.

Purpose

To save the current configuration state of the listener, including trace level, trace file, trace directory, and logging to the `listener.ora` file. Any changes are stored in `listener.ora`, preserving formatting, comments, and case as much as possible. Before modification of the `listener.ora` file, a backup of the file, called `listener.bak`, is created.

Syntax

From the operating system:

```
lsnrctl SAVE_CONFIG listener_name
```

From the Listener Control utility:

```
LSNRCTL> SAVE_CONFIG listener_name
```

Arguments

`listener_name`: The listener name, if the default name of `LISTENER` is not used.

Usage Notes

This command enables you to save all run-time configuration changes to the `listener.ora` file.

Example

```
LSNRCTL> SAVE_CONFIG listener
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)
(PORT=1521)))
Saved LISTENER configuration parameters.
Listener Parameter File   /oracle/network/admin/listener.ora
Old Parameter File   /oracle/network/admin/listener.bak
The command completed successfully
```

1.5.6 SERVICES

Use the Listener Control utility command `SERVICES` to return details about database services, instances, and service handlers to which listeners forward client connection requests.

Purpose

To obtain detailed information about the database services, instances, and service handlers (dispatchers and dedicated servers) to which the listener forwards client connection requests.

Prerequisites

None

Syntax

Arguments

From the operating system:

```
lsnrctl SERVICES listener_name
```

From the Listener Control utility:

```
LSNRCTL> SERVICES listener_name
```

listener_name: The listener name, if the default name of `LISTENER` is not used.

Usage Notes

The `SET DISPLAYMODE` command changes the format and the detail level of the output.

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for a complete description of `SERVICES` output

Example

This example shows `SERVICES` output in the default display mode. The output shows the following:

- An instance named `sales` belonging to two services, `sales1.us.example.com` and `sales2.us.example.com`, with a total of three service handlers.
- Service `sales1.us.example.com` is handled by one dispatcher only.
- Service `sales2.us.example.com` is handled by one dispatcher and one dedicated server, as specified in the following.

```
LSNRCTL> SERVICES
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
```

```
Services Summary...
Service "sales1.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:10000 state:ready
         DISPATCHER <machine: sales-server, pid: 5696>
         (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=53411))
Service "sales2.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 2 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
      "D001" established:0 refused:0 current:0 max:10000 state:ready
         DISPATCHER <machine: sales-server, pid: 5698>
         (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=52618))
The command completed successfully
```

1.5.7 SET

Use the Listener Control utility command `SET` to alter listener parameter values.

Purpose

To alter the parameter values for the listener. Parameter value changes remain in effect until the listener is shut down. To make changes permanent, use the SAVE_CONFIG command to save changes to the `listener.ora` file.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET parameter
```

From the Listener Control utility:

```
LSNRCTL> SET parameter
```

Arguments

`parameter`: A `SET` parameter to modify its configuration setting. Parameters are shown in the example output.

When you enter `SET` without an argument, the Listener Control utility lists all of the parameters.

Usage Notes

If you are using the `SET` commands to alter the configuration of a listener other than the default `LISTENER` listener, then use the `SET CURRENT_LISTENER` command to set the name of the listener to administer.

Example

```
LSNRCTL> SET
The following operations are available with set.
An asterisk (*) denotes a modifier or extended command.
current_listener
displaymode
inbound_connect_timeout
log_file
log_directory
log_status
rawmode
save_config_on_stop
trc_file
trc_directory
trc_level
```

1.5.8 SET CURRENT_LISTENER

Use the Listener Control utility command `SET CURRENT_LISTENER` to set the name of the listener to administer.

Purpose

To set the name of the listener that you want to administer. After you set the listener name with this command, you can issue subsequent commands that normally require *`listener_name`* without specifying the listener.

Syntax

From the Listener Control utility:

```
LSNRCTL> SET CURRENT_LISTENER listener_name
```

Arguments

`listener_name`: The listener name, if you are not using the default name `LISTENER`.

Usage Notes

When you specify a listener name using `SET CURRENT_LISTENER`, the Listener Control utility commands act on the listener name that you specify with this command. You do not have to continue to specify the name of the listener.

Example

```
LSNRCTL> SET CURRENT_LISTENER lsnr
Current Listener is lsnr
```

1.5.9 SET DISPLAYMODE

Use the Listener Control utility command `SET DISPLAYMODE` to change the format and detail level for the `SERVICES` and `STATUS` commands.

Purpose

To change the format and level of detail for the `SERVICES` and `STATUS` commands.

Syntax

From the Listener Control utility:

```
LSNRCTL> SET DISPLAYMODE {compat | normal | verbose | raw}
```

Arguments

Specify one of the following modes:

`compat`: Output that is compatible with earlier releases of the listener.

`normal`: Output that is formatted and descriptive. Oracle recommends this mode.

`verbose`: All data received from the listener in a formatted and descriptive output.

`raw`: All data received from the listener without any formatting. This argument should be used only if recommended by Oracle Support Services.

Example

```
LSNRCTL> SET DISPLAYMODE normal
Service display mode is NORMAL
```

1.5.10 SET INBOUND_CONNECT_TIMEOUT

Use the Listener Control utility command `SET INBOUND_CONNECT_TIMEOUT` to specify the duration in which clients must complete connection requests to listeners after establishing network connections.

Purpose

To specify the time, in seconds, for the client to complete its connect request to the listener after establishing the network connection.

If the listener does not receive the client request in the time specified, then it terminates the connection. In addition, the listener logs the IP address of the client and an `ORA-12525:TNS: listener has not received client's request in time allowed` error message to the `listener.log` file.

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for additional information about specifying the time out for client connections

Syntax

From the Listener Control utility:

```
LSNRCTL> SET INBOUND_CONNECT_TIMEOUT time
```

Arguments

time: The duration of time in seconds. Default setting is 60 seconds.

Example

```
LSNRCTL> SET INBOUND_CONNECT_TIMEOUT 2
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "inbound_connect_timeout" set to 2
The command completed successfully.
```

1.5.11 SET LOG_DIRECTORY

Use the Listener Control utility command `SET LOG_DIRECTORY` to set the destination directory to which the listener log file is written.

Purpose

To set the destination directory to which the listener log file is written. By default, the log file is written to the `ORACLE_HOME/network/log` directory.

> **Note:**
>
> This command works only if Automatic Diagnostic Repository (ADR) is disabled. The default is for ADR to be enabled and to use the log directory `ORACLE_HOME/log/diag/product_type`.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET LOG_DIRECTORY directory
```

From the Listener Control utility:

```
LSNRCTL> SET LOG_DIRECTORY directory
```

Arguments

directory: The directory path of the listener log file.

Example

```
LSNRCTL> SET LOG_DIRECTORY /usr/oracle/admin
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "log_directory" set to /usr/oracle/admin
The command completed successfully
```

1.5.12 SET LOG_FILE

Use the Listener Control utility command `SET LOG_FILE` to set the listener log file name.

Purpose

To set the name for the listener log file. By default, the log file name is `listener.log`.

> **Note:**
>
> This command works only if Automatic Diagnostic Repository (ADR) is disabled. The default is for ADR to be enabled and to use the log directory `ORACLE_HOME/log/diag/product_type`.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET LOG_FILE file_name
```

From the Listener Control utility:

```
LSNRCTL> SET LOG_FILE file_name
```

Arguments

file_name: The file name of the listener log.

Example

```
LSNRCTL> SET LOG_FILE list.log
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "log_file" set to list.log
The command completed successfully
```

1.5.13 SET LOG_STATUS

Use the Listener Control utility command `SET LOG_STATUS` to turn listener logging on or off.

Purpose

To turn listener logging on or off.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET LOG_STATUS {on | off}
```

From the Listener Control utility:

```
LSNRCTL> SET LOG_STATUS {on | off}
```

Arguments

`on`: To turn logging on.

`off`: To turn logging off.

Example

```
LSNRCTL> SET LOG_STATUS on
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "log_status" set to ON
The command completed successfully
```

1.5.14 SET SAVE_CONFIG_ON_STOP

Use the Listener Control utility command `SET SAVE_CONFIG_ON_STOP` to specify whether listener parameter value changes that you make with the `SET` command are saved to the `listener.ora` file when you stop the listener with the `STOP` command.

Purpose

To specify whether changes made to parameter values for the listener by the `SET` command are saved to the `listener.ora` file when the listener is stopped with the `STOP` command.

When you save changes, the Listener Control utility tries to preserve formatting, comments, and letter case. Before the command modifies the `listener.ora` file, it creates a backup of the file called `listener.bak`.

To have all parameters saved immediately, use the `SAVE_CONFIG` command.

Syntax

From the operating system:

```
lsnrctl SET SAVE_CONFIG_ON_STOP  {on | off}
```

From the Listener Control utility:

```
LSNRCTL> SET SAVE_CONFIG_ON_STOP  {on | off}
```

Arguments

`on`: To save configuration to `listener.ora`.

`off`: To not save configuration to `listener.ora`.

Example

```
LSNRCTL> SET SAVE_CONFIG_ON_STOP on
LISTENER parameter "save_config_on_stop" set to ON
The command completed successfully
```

1.5.15 SET TRC_DIRECTORY

Use the Listener Control utility command `SET TRC_DIRECTORY` to set the destination directory into which Oracle writes listener trace files.

Purpose

To set the destination directory where the listener trace files are written. By default, the trace file are written to the `ORACLE_HOME/network/trace` directory.

> **Note:**
>
> This command works only if Automatic Diagnostic Repository (ADR) is disabled. The default is for ADR to be enabled and to use the log directory `ORACLE_HOME/log/diag/`*`product_type`*.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET TRC_DIRECTORY directory
```

From the Listener Control utility:

```
LSNRCTL> SET TRC_DIRECTORY directory
```

Arguments

directory: The directory path of the listener trace files.

Example

```
LSNRCTL> SET TRC_DIRECTORY /usr/oracle/admin
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "trc_directory" set to /usr/oracle/admin
The command completed successfully
```

1.5.16 SET TRC_FILE

Use the Listener Control utility command `SET TRC_FILE` to set the names of listener trace files.

Purpose

To set the name of the listener trace file. By default, the trace file name is `listener.trc`.

> **Note:**
>
> This command works only if Automatic Diagnostic Repository (ADR) is disabled. The default is for ADR to be enabled and to use the log directory `ORACLE_HOME/log/diag/product_type`.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET TRC_FILE file_name
```

From the Listener Control utility:

```
LSNRCTL> SET TRC_FILE file_name
```

Arguments

file_name: The file name of the listener trace.

Example

```
LSNRCTL> SET TRC_FILE list.trc
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
LISTENER parameter "trc_file" set to list.trc
The command completed successfully
```

1.5.17 SET TRC_LEVEL

Use the Listener Control utility command `SET TRC_LEVEL` to set a specific listener tracing level.

Purpose

To set a specific level of tracing for listeners.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SET TRC_LEVEL level
```

From the Listener Control utility:

```
LSNRCTL> SET TRC_LEVEL level
```

Arguments

level: One of the following trace levels:

- `off` for no trace output
- `user` for user trace information
- `admin` for administration trace information
- `support` for Oracle Support Services trace information

Usage Notes

This command has the same functionality as the `TRACE` command.

Example

```
LSNRCTL> SET TRC_LEVEL admin
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
```

```
LISTENER parameter "trc_level" set to admin
The command completed successfully
```

1.5.18 SHOW

Use the Listener Control utility command `SHOW` to list current listener parameter values.

Purpose

To view the current parameter values for the listener.

All of the `SET` parameters have equivalent `SHOW` parameters.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SHOW parameter
```

From the Listener Control utility:

```
LSNRCTL> SHOW parameter
```

Arguments

`parameter`: A parameter whose settings you want to review. Parameters are shown in the example output.

When you enter `SHOW` without an argument, the Listener Control utility lists all the parameters.

Example

```
LSNRCTL> SHOW
The following properties are available with SHOW:
An asterisk (*) denotes a modifier or extended command:
current_listener
displaymode
inbound_connect_timeout
log_file
log_directory
log_status
rawmode
save_config_on_stop
trc_file
trc_directory
trc_level
```

1.5.19 SHOW STATS

Use the Listener Control utility `SHOW STATS` command with the `-REG` option to display statistics about database service registration commands.

Purpose

To display statistics about the number of registration commands that the database listener receives while handling client connection requests.

Using these statistics, you can monitor service registration or service update operations, such as `REGISTER`, `UPDATE`, `RE-REGISTER`, and `UN-REGISTER`. This also helps you evaluate the traffic and overhead of these operations at Oracle Database.

Prerequisites

The listener must be running.

Syntax

From the operating system:

```
lsnrctl SHOW STATS -REG
```

From the Listener Control (`LSNRCTL`) utility:

```
LSNRCTL> SHOW STATS -REG
```

Usage Notes

- In addition to `LSNRCTL`, you can run this command from the Oracle Connection Manager Control (`CMCTL`) utility.
- If you enter `SHOW STATS -REG` without any argument, then the output displays a global-level data for all instances, registered services, handlers allocated to each service, listening endpoints, and access control lists (ACLs).
- Recent count is a periodic count of all commands received by the listener from the last reset, that is, from the time you cleared the `Recent` section using `-clear`. If you have not used `-clear` till now, then this field displays a cumulative count (total number of commands collected since the listener started).

  For a detailed description about other sections of the output, see *Oracle Database Net Services Administrator's Guide*.

Arguments and Examples

Argument and Description	Example

To display a recent and cumulative count of all registration commands at a global level:
`-reg`
	`LSNRCTL:lsnr1> show stats -reg`
	`------------------------`
	`Global Level:`
	`Recent`
	`Recent Duration: 5 days 17 hr. 15 min. 25 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 2 2 2 4 0`
	`Updates 3 0 0 12 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 3 3 3 6 0`
	`Updates 4 0 0 12 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	`The command completed successfully`

To display a recent and cumulative count of all registration commands for the specified instance name:
`-inst` *`instance_name`*
	`LSNRCTL:lsnr1> show stats -reg -inst sales1`
	`------------------------`
	`Instance Name: sales1`
	`Recent`
	`Recent Duration: 5 days 15 hr. 17 min. 18 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 2 2 2 4 0`
	`Updates 3 0 0 12 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 2 2 2 4 0`
	`Updates 3 3 0 12 0`
	`Re-Register 0 0 3 0 0`
	`Un-Register 0 0 0 0 0`
	`The command completed successfully`

Argument and Description	Example

To display a recent and cumulative count of registration commands for all instances:
`-all_inst`
	`LSNRCTL:lsnr1> show stats -reg -all_inst`
	`------------------------`
	`Instance Name: sales1`
	`Recent`
	`Recent Duration: 5 days 15 hr. 3 min. 2 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 1 1 1 2 0`
	`Updates 21 0 0 5 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 1 1 1 4 0`
	`Updates 25 0 2 5 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	
	`------------------------`
	`Instance Name: sales2`
	`Recent`
	`Recent Duration: 2 days 5 hr. 3 min. 2 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 1 1 1 2 0`
	`Updates 2 0 0 3 0`
	`Re-Register 0 1 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 1 1 1 3 0`
	`Updates 10 0 2 5 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	
	`The command completed successfully`

To display a recent and cumulative count of all registration commands for the specified service name:
`-serv` *`service_name`*
	`LSNRCTL:lsnr1> show stats -reg -serv sales.us.example.com`
	`------------------------`
	`Service Name: sales.us.example.com`
	`Recent Duration: 5 days 15 hr. 4 min. 10 sec`
	`Flags Goodness Delta`
	`Recent 2 0 0`
	`Cumulative 0 0 0`
	`The command completed successfully`

Argument and Description	Example

To display a recent and cumulative count of registration commands for all database services:
`-all_serv`
	`LSNRCTL:lsnr1> show stats -reg -all_serv`
	`------------------------`
	`Service Name: sales.us.example.com`
	`Recent Duration: 5 days 15 hr. 5 min. 25 sec`
	`Flags Goodness Delta`
	`Recent 1 2 0`
	`Cumulative 2 0 0`
	`------------------------`
	`Service Name: employee.us.example.com`
	`Recent Duration: 8 days 2 hr. 5 min. 25 sec`
	`Flags Goodness Delta`
	`Recent 1 0 0`
	`Cumulative 1 0 0`
	`The command completed successfully`

To reset the `Recent` section to `0` after fetching the values:
`-clear`
This gives you a fresh set of data in the `Recent` section. You can use `-clear` with all the `SHOW STATS` arguments.
	`LSNRCTL:lsnr1> show stats -reg -clear`
	`------------------------`
	`Global Level:`
	`Recent`
	`Recent Duration: 0 days 0 hr. 0 min. 4 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 0 0 0 0 0`
	`Updates 0 0 0 0 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 3 3 3 6 0`
	`Updates 4 0 0 13 0`
	`Re-Register 1 0 0 0 0`
	`Un-Register 0 1 0 0 0`
	`The command completed successfully`

Related Topics

- *Oracle Database Net Services Administrator's Guide*

1.5.20 SPAWN

Use the Listener Control utility command `SPAWN` to start a program that is stored on the computer on which the listener is running and that is listed with an alias in the `listener.ora` file.

Purpose

To start a program stored on the computer on which the listener is running and that is listed with an alias in the `listener.ora` file.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl SPAWN listener_name alias (arguments='arg1,arg2,...')
```

From the Listener Control utility:

```
LSNRCTL> SPAWN listener_name alias (arguments='arg1,arg2,...')
```

Arguments

listener_name: The listener name, if the default name of `LISTENER` is not used.

alias: The alias of the program to be spawned is specified by a `listener.ora` file entry that is similar to the following:

```
alias = (PROGRAM=(NAME=)(ARGS=)(ENVS=))
```

For example:

```
nstest = (PROGRAM=(NAME=nstest)(ARGS=test1)(ENVS='ORACLE_HOME=/usr/
oracle'))
```

Example

The `nstest` program, shown in the preceding section, can then be spawned using the following command:

```
lsnrctl SPAWN listener_name nstest
```

1.5.21 START

Use the Listener Control utility command `START` to start a named listener.

Purpose

To start the named listener.

Prerequisites

The listener must not be running.

Syntax

From the operating system:

```
lsnrctl START listener_name
```

From the Listener Control utility:

```
LSNRCTL> START listener_name
```

> **Note:**
>
> On Microsoft Windows, if the database was installed with the Oracle Home User, then the utility can prompt for a password. The password is the operating system password for the Oracle Home User. The prompt is displayed only if the listener service does not exist and if it needs to be created as part of starting the listener.

Arguments

listener_name: The listener name if the default name of `LISTENER` is not used.

Usage Notes

To start a listener that you configured in the `listener.ora` file whose name does not contain the string `LISTENER`.

For example, if the listener name is `tcp_lsnr`, enter:

```
lsnrctl START tcp_lsnr
```

From the Listener Control utility:

```
LSNRCTL> START tcp_lsnr
```

Example

```
LSNRCTL> START

Starting /private/sales_group/sales/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 23.4.0.0.0
System parameter file is $ORACLE_HOME/network/admin/listener.ora
Log messages written to $ORACLE_BASE/diag/tnslsnr/node_name/listener/alert/
log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)
(PORT=1521)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)
(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 23.4.0.0.0
Start Date                21-MAR-2024 21:50:49
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authetication
```

```
SNMP                       OFF
Listener Parameter File  $ORACLE_HOME/network/admin/listener.ora
Listener Log File        $ORACLE_BASE/diag/tnslsnr/node_name/listener/
alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
The listener supports no services
The command completed successfully
```

> **See Also:**
>
> *Oracle Database Administrator's Reference for Microsoft Windows* for information about the Oracle Home User

1.5.22 STATUS

Use the Listener Control utility command `STATUS` to show listener status information.

Purpose

To display basic status information about a listener, including a summary of listener configuration settings, listening protocol addresses, and a summary of services that are registered with the listener.

> **Note:**
>
> You can also obtain the status of the listener through the Oracle Enterprise Manager Cloud Control console.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl STATUS listener_name
```

From the Listener Control utility:

```
LSNRCTL> STATUS listener_name
```

Arguments

`listener_name`: The listener name, if the default name of `LISTENER` is not used.

Usage Notes

The `SET DISPLAYMODE` command changes the format and the level of output detail.

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for a complete description of `STATUS` output

Example

The following example shows `STATUS` output in the default display mode. The output contains:

- Listener configuration settings
- Listening endpoints summary
- Services summary, which is an abbreviated version of the `SERVICES` command output

```
LSNRCTL> STATUS
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 23.4.0.0.0 -
Production
Start Date                12-MAR-2024 12:02:00
Uptime                    0 days 0 hr. 5 min. 29 sec
Trace Level               support
Security                  OFF
SNMP                      OFF
Listener Parameter File   /oracle/network/admin/listener.ora
Listener Log File         /oracle/network/log/listener.log
Listener Trace File       /oracle/network/trace/listener.trc

Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=2484)))

Services Summary...
Service "sales1.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 1 handler(s) for this service...
Service "sales2.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 2 handler(s) for this service...
The command completed successfully
```

1.5.23 STOP

Use the Listener Control utility command `STOP` to stop the named listener.

Purpose

To stop the named listener.

Prerequisites

The named listener must be running.

Syntax

From the operating system:

```
lsnrctl STOP listener_name
```

From the Listener Control utility:

```
LSNRCTL> STOP listener_name
```

Arguments

`listener_name`: The listener name, if the default name of `LISTENER` is not used.

Example

```
LSNRCTL> STOP
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
The command completed successfully
```

1.5.24 TRACE

Use the Listener Control utility command `TRACE` to set listener tracing.

Purpose

To set tracing for the listener.

Syntax

From the operating system:

```
lsnrctl trace level listener_name
```

From the Listener Control utility:

```
LSNRCTL> trace level listener_name
```

Arguments

level: One of the following trace levels:

- `off` for no trace output
- `user` for user trace information
- `admin` for administration trace information
- `support` for Oracle Support Services trace information

listener_name: Specify the listener name, if the default name of `LISTENER` is not used.

Usage Notes

This command has the same functionality as the `SET TRC_LEVEL` command.

Example

```
LSNRCTL> TRACE ADMIN lsnr
Connecting to (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
Opened trace file: /oracle/network/trace/listener.trc
The command completed successfully
```

1.5.25 VERSION

Use the Listener Control utility command `VERSION` to show the current version of Listener Control utility.

Purpose

To display the current version of Listener Control utility.

Prerequisites

None

Syntax

From the operating system:

```
lsnrctl VERSION listener_name
```

From the Listener Control utility:

```
LSNRCTL> VERSION listener_name
```

Arguments

listener_name: The listener name, if the default name of `LISTENER` is not used.

Example

```
LSNRCTL> version listener
Connecting to ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
TNSLSNR for Linux: Version 23.4.0.0.0
        TNS for Linux: Version 23.4.0.0.0
        Oracle Bequeath NT Protocol Adapter for Linux: Version
23.4.0.0.0
        Unix Domain Socket IPC NT Protocol Adaptor for Linux: Version
23.4.0.0.0
        TCP/IP NT Protocol Adapter for Linux: Version 23.4.0.0.0
The command completed successfully
```

2

Oracle Connection Manager Control Utility

This chapter describes Oracle Connection Manager Control Utility commands and syntax .

- Connection Manager Control Utility Command Modes and Syntax
  Oracle Connection Manager Control utility (CMCTL) enables you to start up, configure, and alter how client connection requests are managed.
- Oracle Connection Manager Control Utility Overview
- Oracle Connection Manager Control Utility Commands
  Use Oracle Connection Manager Control utility commands to manage and configure Oracle Connection Manager instances.

2.1 Connection Manager Control Utility Command Modes and Syntax

Oracle Connection Manager Control utility (CMCTL) enables you to start up, configure, and alter how client connection requests are managed.

The syntax of the Oracle Connection Manager Control utility is:

```
cmctl command [argument]
```

Oracle Connection Manager Control utility supports the following command types:

- Initialization and termination commands such as `STARTUP` and `SHUTDOWN`
- Alter commands such as `SET LOG_LEVEL` and `SET EVENT`
- Display commands, such as `SHOW STATUS` and `SHOW RULES`
- Gateway commands such as `SHOW GATEWAYS` and `RESUME GATEWAYS`

Note:

Use `SET` commands to dynamically alter configuration parameters that control how the listener receives client connections. Changes only remain in effect until Oracle Connection Manager shuts down. You cannot save changes to the `cman.ora` file.

Use the Oracle Connection Manager Control utility in command mode or batch mode.

- Using command mode:
  - From the Oracle Connection Manager Control utility:

    Enter `cmctl` at the command line to obtain the program prompt and run the command:

    ```
    cmctl
    CMCTL> command
    ```

  - From the operating system:

    Enter the command from the operating system command prompt:

    ```
    cmctl [command] [argument1 . . . argumentN] [-c
    instance_name]
    ```

    You can append an Oracle Connection Manager instance name as an argument to all commands that you run in this manner. If you do not include an Oracle Connection Manager instance name, then the default instance name is assumed. The default name is `cman_hostname`.
- Using batch mode:

  You can combine commands in a standard text file and run them as a sequence of commands. To run commands in batch mode, use the following syntax:

  ```
  cmctl @input_file
  ```

See Also:

Oracle Database Net Services Administrator's Guide for more information about Oracle Connection Manager architecture

2.2 Oracle Connection Manager Control Utility Overview

Oracle Connection Manager Control utility (CMCTL) enables you to administer Oracle Connection Manager. Use Oracle Connection Manager Control utility commands to

administer one or more Oracle Connection Manager instances. Additionally, you can view and change parameter settings.

2.3 Oracle Connection Manager Control Utility Commands

Use Oracle Connection Manager Control utility commands to manage and configure Oracle Connection Manager instances.

- ADMINISTER
  Use the Oracle Connection Manager Control Utility command `ADMINISTER` to select Oracle Connection Manager instances to administer.
- CLOSE CONNECTIONS
  Use the Oracle Connection Manager Control Utility command `CLOSE CONNECTIONS` to terminate connections.
- CLOSE NON_ADMIN_ENDPOINTS
  Use the Oracle Connection Manager Control Utility command `CLOSE NON_ADMIN_ENDPOINTS` to close all non-admin listening endpoints.
- EXIT
  Use the Oracle Connection Manager Control Utility command `EXIT` to exit from Oracle Connection Manager utility.
- HELP
  Use the Oracle Connection Manager Control Utility `HELP` command to display all of the Oracle Connection Manager Control utility commands or to show the syntax of a particular command.
- QUIT
  Use the Oracle Connection Manager Control Utility command `QUIT` to exit Oracle Connection Manager Control utility.
- RELOAD
  Use the Oracle Connection Manager Control utility `RELOAD` command to make the utility dynamically reread parameters and rules.
- RESUME GATEWAYS
  Use the Oracle Connection Manager Control Utility command `RESUME GATEWAYS` to resume suspended gateway processes.
- SET
  Use the Oracle Connection Manager Control Utility `SET` command to list the parameters that you can modify using this command.
- SET ASO_AUTHENTICATION_FILTER
  Use the Oracle Connection Manager Control Utility command `SET ASO_AUTHENTICATION_FILTER` to indicate whether clients must use Oracle Database security authentication.
- SET CONNECTION_STATISTICS
  Use the Oracle Connection Manager Control Utility `SET CONNECTION_STATISTICS` command to specify whether gateway processes collect connection statistics.
- SET EVENT
  Use the Oracle Connection Manager Control Utility `SET EVENT` command to log information for a particular event.

- SET IDLE_TIMEOUT
  Use the Oracle Connection Manager Control Utility `SET IDLE_TIMEOUT` command to specify the amount of time that clients can be idle without transmitting data.
- SET INBOUND_CONNECT_TIMEOUT
  Use the Oracle Connection Manager Control Utility `SET INBOUND_CONNECT_TIMEOUT` command to specify the maximum amount of time that Oracle Connection Manager listeners wait for client connection requests before timing out.
- SET LOG_DIRECTORY
  Use the Oracle Connection Manager Control Utility `SET LOG_DIRECTORY` command to designate where Oracle Connection Manager log files are written.
- SET LOG_LEVEL
  Use the Oracle Connection Manager Control Utility `SET LOG_LEVEL` command to set Oracle Connection Manager log levels.
- SET OUTBOUND_CONNECT_TIMEOUT
  Use the Oracle Connection Manager Control Utility `SET OUTBOUND_CONNECT_TIMEOUT` command to specify the time limit that Oracle Connection Manager instances wait for server connections before timing out.
- SET SESSION_TIMEOUT
  Use the Oracle Connection Manager Control utility `SET SESSION_TIMEOUT` command to specify the maximum duration of Oracle Connection Manager sessions.
- SET TRACE_DIRECTORY
  Use the Oracle Connection Manager Control utility `SET TRACE_DIRECTORY` command to designate where Oracle Connection Manager instance trace files are written.
- SET TRACE_LEVEL
  Use the Oracle Connection Manager Control utility `SET TRACE_LEVEL` command to set Oracle Connection Manager instance trace levels.
- SHOW
  Use the Oracle Connection Manager Control utility `SHOW` command to display the parameters that you can use as arguments for this command.
- SHOW ALL
  Use the Oracle Connection Manager Control utility `SHOW ALL` command to combine and display `SHOW PARAMETERS` and `SHOW RULES` command output.
- SHOW CONNECTIONS
  Use the Oracle Connection Manager Control utility `SHOW CONNECTIONS` command to display connection information.
- SHOW DEFAULTS
  Use the Oracle Connection Manager Control utility `SHOW DEFAULTS` command to display default parameter settings.
- SHOW EVENTS
  Use the Oracle Connection Manager Control utility `SHOW EVENTS` command to display events that are currently operating.
- SHOW GATEWAYS
  Use the Oracle Connection Manager Control utility `SHOW GATEWAYS` command to display the statuses of gateway processes.

- SHOW PARAMETERS
  Use the Oracle Connection Manager Control utility `SHOW PARAMETERS` command to display the parameter settings for an instance.
- SHOW RULES
  Use the Oracle Connection Manager Control Utility `SHOW RULES` command to display an instance access control list.
- SHOW SERVICES
  Use the Oracle Connection Manager Control utility `SHOW SERVICES` command to display Oracle Connection Manager instance information.
- SHOW STATS
  Use the Oracle Connection Manager Control utility `SHOW STATS` command with the `-REG` option to display statistics about database service registration commands.
- SHOW STATUS
  Use the Oracle Connection Manager Control utility `SHOW STATUS` command to display Oracle Connection Manager instance information.
- SHOW VERSION
  Use the Oracle Connection Manager Control utility `SHOW VERSION` command
- SHUTDOWN
  Use the Oracle Connection Manager Control utility `SHUTDOWN` command to shut down gateway processes or an entire Oracle Connection Manager instance.
- STARTUP
  Use the Oracle Connection Manager Control utility `STARTUP` command to start Oracle Connection Manager.
- SUSPEND GATEWAY
  Use the Oracle Connection Manager Control utility `SUSPEND GATEWAY` command to specify the gateway processes than cannot accept new client connections.

2.3.1 ADMINISTER

Use the Oracle Connection Manager Control Utility command `ADMINISTER` to select Oracle Connection Manager instances to administer.

Purpose

To select an Oracle Connection Manager instance.

Prerequisites

None

Syntax

From the Oracle Connection Manager Control utility:

```
CMCTL> ADMINISTER [-c] instance_name
```

Arguments

instance_name: The Oracle Connection Manager instance name that you want to administer. Instances are defined in the `cman.ora` file.

Usage Notes

You can run the `ADMINISTER` command only within the utility. You cannot issue this command from the operating system.

`ADMINISTER` enables you to choose Oracle Connection Manager instances to administer. To start the Oracle Connection Manager instance, run the `STARTUP` command.

When you omit the instance name from the command, the instance that is administered defaults to the local instance.

Use the `-c` option to administer an instance that is not the local instance.

Example

```
CMCTL> ADMINISTER cman_indl040ad
Current instance cman_indl040ad is already started
Connections refer to (address=(protocol=TCP)(host=indl040ad)
(port=1560)).
The command completed successfully
```

2.3.2 CLOSE CONNECTIONS

Use the Oracle Connection Manager Control Utility command `CLOSE CONNECTIONS` to terminate connections.

Purpose

To terminate connections, using specific qualifiers to select the connections to close.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl CLOSE CONNECTIONS [in state] [gt time] [from source] [to
destination]
[for service] [using gateway_process_id] [connect_identifier_list]
[-c cman_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> CLOSE CONNECTIONS [in state] [gt time] [from source] [to
destination]
[for service] [using gateway_process_id] [connect_identifier_list
```

Arguments

state: One of the following values to specify the connection state:

- `idle`: Connections that are inactive in the established state.
- `connecting`: Connections that are in the process of connecting.
- `established`: Connections that are connected and are transferring data.
- `terminating`: Connections that are disconnecting.

If you do not specify a state, then `CLOSE CONNECTIONS` defaults to all possible states. If the time qualifier is included under these conditions, then the time specified is the amount of time that has elapsed since a client initiated a connection.

time: The time format. Use the following format to specify connections that have a duration greater than the time indicated:

`gt[`*`hh`*`:`*`mm`*`:]`*`ss`*

source: The source address. Use one of the following formats to specify the source address:

- `from` *`IP`*
- `from` *`hostname`*
- `from` *`subnet`*

destination: The destination address. Use one of the following formats to specify the destination address:

- `to` *`IP`*
- `to` *`hostname`*
- `to` *`subnet`*

service: The service name. Use the `service_name` parameter to specify the service, such as `sales.us.example.com`.

gateway_process_id: The gateway process identifier is a number. Use this number to specify connections that are proxied by the gateway process indicated. To determine the gateway process identifier, use the Oracle Connection Manager control utility `show gateways` command.

connect_identifier_list: The connection identifiers. Use a space between multiple connection identifiers in a list.

Usage Notes

Because the `CLOSE CONNECTIONS` command terminates connections, it might generate error messages on both the client and the server sides.

The `IDLE` state qualifier always requires a time qualifier.

Issuing `CLOSE CONNECTIONS` without an argument closes all connections.

Examples

The following example shuts down connections in any state. The elapsed time of the connection must be greater than 1 hour and 30 minutes. The connection source is the specified subnet, and the destination is the specified host name.

```
CMCTL> CLOSE CONNECTIONS gt 1:30:00 from 192.0.2.32/24 to host1
```

The following example shuts down those connections proxied by gateway process `0` that have been in the idle state more than 30 minutes:

```
CMCTL> CLOSE idle CONNECTIONS gt 30:00 using 0
```

The following example shuts down connections that are connected to the service `sales.us.example.com`:

```
CMCTL> CLOSE established CONNECTIONS for sales.us.example.com
```

REST API for CLOSE CONNECTIONS Command

```
POST /close/connections
JSON Payload
{
“in” : [ “idle” | “connecting” | “established” | “terminating”]
 “gt” : “[hh:mm:]ss”,
  “from” : [“source ip “ | “hostname “ | “subnet”],
  “to” : [“destination ip” | “hostname” | “subnet”],
  “for” : “service name”,
  “using” : “gateway process identifier”,
  “connect_id_list” : [id1, id2, .. ]
}
```

2.3.3 CLOSE NON_ADMIN_ENDPOINTS

Use the Oracle Connection Manager Control Utility command `CLOSE NON_ADMIN_ENDPOINTS` to close all non-admin listening endpoints.

Purpose

To shut down all non-admin listening addresses.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl close non_admin_endpoints [-c instance_name]
```

From Oracle Connection Manager Control utility:

```
CMCTL> CLOSE NON_ADMIN_ENDPOINTS
```

Usage Notes

You can tag Oracle Connection Manager addresses as admin endpoints by specifying the `ADDRESS` networking parameter. Thus, when you run the `CLOSE NON_ADMIN_ENDPOINTS` command, Oracle Connection Manager Control utility continues to run admin commands using tagged listening endpoints. The closed endpoints can be used by other process or another instance of Oracle Connection Manager.

Example

```
CMCTL> CLOSE NON_ADMIN_ENDPOINTS
The command completed successfully.
```

Related Topics

- ADDRESS
  The `ADDRESS` networking parameter specifies the protocol address of Oracle Connection Manager.

2.3.4 EXIT

Use the Oracle Connection Manager Control Utility command `EXIT` to exit from Oracle Connection Manager utility.

Purpose

To exit from Oracle Connection Manager Control utility.

Prerequisites

None

Syntax

From the operating system:

`cmctl EXIT [-c instance_name]`
From Oracle Connection Manager Control utility:

```
CMCTL> EXIT
```

Usage Notes

This command is identical to the `QUIT` command.

Example 2-1 Example

```
CMCTL> EXIT
```

2.3.5 HELP

Use the Oracle Connection Manager Control Utility `HELP` command to display all of the Oracle Connection Manager Control utility commands or to show the syntax of a particular command.

Purpose

To provide a list of all commands for Oracle Connection Manager Control utility or to provide help with the syntax of a particular command.

Prerequisites

None

Syntax

From the operating system:

```
cmctl HELP [command] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> HELP [command]
```

From the operating system:

```
cmctl HELP [command] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> HELP [command]
```

Arguments

command: Specify a `HELP` command. Commands are shown in the following sample output.

When you enter a command as an argument to `HELP`, Oracle Connection Manager Control utility displays information about how to use the command. When you enter `HELP` without an argument, Oracle Connection Manager Control utility displays a list of all the commands.

Example

```
CMCTL> HELP
The following operations are available
An asterisk (*) denotes a modifier or extended command:

administer      close*          exit            reload
resume*         save_passwd     set*            show*
shutdown        sleep           startup         suspend*
show_version    quit
```

2.3.6 QUIT

Use the Oracle Connection Manager Control Utility command `QUIT` to exit Oracle Connection Manager Control utility.

Purpose

To exit Oracle Connection Manager Control utility and return to the operating system prompt.

Prerequisites

None

Syntax

```
CMCTL> QUIT
```

From the Oracle Connection Manager Control utility:

```
cmctl QUIT
```

From the operating system:

Usage Notes

This command is identical to the `EXIT` command.

Example

```
CMCTL> QUIT
```

2.3.7 RELOAD

Use the Oracle Connection Manager Control utility `RELOAD` command to make the utility dynamically reread parameters and rules.

Purpose

To dynamically reread parameters and rules.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl RELOAD [-with_ha] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> RELOAD [-with_ha]
```

Arguments

`-with_ha`: It is used to reload `cman.ora` without dropping registrations

Usage Notes

Configuration information that you modify using the `RELOAD` command applies only to new connections. Existing connections are unaffected. The `SET RELOAD` command restores configurations set in `cman.ora`, and overrides the `SET` command.

The `RELOAD` command reregisters gateways with the Oracle Connection Manager listener during which some new connections might be refused until the registration completes.

You can use the `-with_ha` option with `RELOAD` to not drop registrations, thus providing high service availability during reload.

Example

```
CMCTL> RELOAD
The command completed successfully
```

REST API for RELOAD Command

```
POST /reload
```

2.3.8 RESUME GATEWAYS

Use the Oracle Connection Manager Control Utility command `RESUME GATEWAYS` to resume suspended gateway processes.

Purpose

To resume gateway processes that have been suspended.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

`cmctl RESUME GATEWAYS [gateway_process_id] [cman_name]`

From the Oracle Connection Manager Control utility:

`CMCTL> RESUME GATEWAYS [gateway_process_id]`

Arguments

gateway_process_id: One or more gateway processes to reopen. Separate multiple gateway processes using a space between the process identifiers.

Usage Notes

Running the `RESUME GATEWAYS` command without an argument reopens all closed gateway processes.

Example

```
CMCTL> RESUME GATEWAYS 1
The command completed successfully
```

REST API for RESUME GATEWAYS Command

```
POST /resume/gateways
JSON Payload
   {
    “gateway_id_list” : [id1, id2, .. ]
   }
```

2.3.9 SET

Use the Oracle Connection Manager Control Utility `SET` command to list the parameters that you can modify using this command.

Purpose

To display a list of parameters that can be modified using this command.

Prerequisites

None

Syntax

From the operating system:

```
cmctl SET
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET
```

Example

```
CMCTL> SET
The following operations are available after set
An asterisk (*) denotes a modifier or extended command:

aso_authentication_filter     outbound_connect_timeout
connection_statistics         session_timeout
event                         trace_directory
idle_timeout                  trace_level
inbound_connect_timeout
log_directory
log_level
```

2.3.10 SET ASO_AUTHENTICATION_FILTER

Use the Oracle Connection Manager Control Utility command `SET ASO_AUTHENTICATION_FILTER` to indicate whether clients must use Oracle Database security authentication.

Purpose

To indicate whether the client must use Oracle Database security to authenticate.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET ASO_AUTHENTICATION_FILTER {on | off}[-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET ASO_AUTHENTICATION_FILTER {on | off}
```

Arguments

`on`: To reject connections that are not using Secure Network Service (SNS) to perform client authentication. SNS is part of Oracle Database security.

`off`: To specify that no authentication is required for client connections. This is the default.

Example

```
CMCTL> set aso_authentication_filter ON
CMAN_user.us.example.com parameter aso_authentication_filter set to ON
The command completed successfully
```

REST API for SET ASO_AUTHENTICATION_FILTER Command

```
POST /set/authlevel
JSON Payload
{
  "authlevel": "true"|"false"
}
```

2.3.11 SET CONNECTION_STATISTICS

Use the Oracle Connection Manager Control Utility `SET CONNECTION_STATISTICS` command to specify whether gateway processes collect connection statistics.

Purpose

To specify whether gateway processes collect connection statistics.

Prerequisites

To specify whether gateway processes collect connection statistics.

Syntax

From the operating system:

```
cmctl SET CONNECTION_STATISTICS {yes | no}[-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET CONNECTION_STATISTICS {yes | no}
```

Arguments

`yes`: To have gateway processes collect connection statistics.

`no`: To not have gateway processes collect connection statistics. This is the default.

Usage Notes

If you `SET CONNECTION_STATISTICS` to `yes`, then you can obtain statistics by running the `SHOW CONNECTIONS` command.

Example

```
CMCTL> set connection_statistics ON
CMAN_user.us.example.com parameter connection_statistics set to ON
The command completed successfully
```

REST API for SET CONNECTION_STATISTICS Command

```
POST /set/connstats/
JSON Payload
{
“connection_statistics” : “yes”|”no”
}
```

2.3.12 SET EVENT

Use the Oracle Connection Manager Control Utility `SET EVENT` command to log information for a particular event.

Purpose

To log information for a particular event.

Syntax

From the operating system:

```
cmctl SET EVENT event_group [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET EVENT event_group {on | off}
```

Arguments

event_group: Specify one of the following event groups:

- `init_and_term`: Initialization and termination event group.
- `memory_ops`: Memory operations event group.
- `conn_hdlg`: Connection handling event group.
- `proc_mgmt`: Process management event group.
- `reg_and_load`: Registration and load update event group.
- `wake_up`: Events related to Connection Manager Administration (CMADMIN) wakeup queue event group.
- `timer`: Gateway timeouts event group.
- `cmd_proc`: Command processing event group.
- `relay`: Events associated with connection control blocks event group.

`on`: To enable an event group.

`off`: To disable an event group.

Usage Notes

The `SET EVENT` command accepts only one argument at a time. To log multiple events, run the command for each event separately.

Example

```
CMCTL> set event memory_ops off
cman11 event memory_ops set to OFF.
The command completed successfully.
```

2.3.13 SET IDLE_TIMEOUT

Use the Oracle Connection Manager Control Utility `SET IDLE_TIMEOUT` command to specify the amount of time that clients can be idle without transmitting data.

Purpose

To specify the amount of time a client can be idle without transmitting data.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET IDLE_TIMEOUT [time] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET IDLE_TIMEOUT [time]
```

Arguments

time: Specify the idle timeout in seconds. The default is 0 (zero), which disables this feature.

Example

```
CMCTL> SET IDLE_TIMEOUT 30
CMAN_user.us.example.com parameter idle_timeout set to 30
The command completed successfully
```

REST API for SET IDLE_TIMEOUT Command

```
POST /set/maxidletime
JSON Payload
{
"idle_timeout" : "time"
}
```

2.3.14 SET INBOUND_CONNECT_TIMEOUT

Use the Oracle Connection Manager Control Utility `SET INBOUND_CONNECT_TIMEOUT` command to specify the maximum amount of time that Oracle Connection Manager listeners wait for client connection requests before timing out.

Purpose

To specify the maximum amount of time the Oracle Connection Manager listener waits for a valid connection request from the client before timing out.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET INBOUND_CONNECT_TIMEOUT [time] [-c instance_name]
```

From Oracle Connection Manager Control:

```
CMCTL> SET INBOUND_CONNECT_TIMEOUT [time]
```

Arguments

time: The inbound connect timeout in seconds. The default is 0 (zero), which disables this feature.

Example

```
CMCTL> SET INBOUND_CONNECT_TIMEOUT 30
CMAN_user.us.example.com parameter inbound_connect_timeout set to 30
The command completed successfully
```

REST API for SET INBOUND_CONNECT_TIMEOUT Command

```
POST /set/maxcntdtime
JSON Payload
{
“inbound_connect_timeout” : “time”
}
```

2.3.15 SET LOG_DIRECTORY

Use the Oracle Connection Manager Control Utility `SET LOG_DIRECTORY` command to designate where Oracle Connection Manager log files are written.

> **Note:**
>
> This command works only if you have not enabled Automatic Diagnostic Repository (ADR). The default is for ADR to be enabled and use the log directory `ORACLE_HOME/log`.

Purpose

To designate where the log files for Oracle Connection Manager are written.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET LOG_DIRECTORY [directory_path] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET LOG_DIRECTORY [directory_path]
```

Arguments

directory_path: The location of the log directory. The default path is as follows:

- Linux and UNIX:

  `ORACLE_HOME/network/log directory`

- Microsoft Windows:

  `ORACLE_HOME\network\log directory`

Usage Notes

Use the `SHOW PARAMETERS` command to determine the location of the log files.

Example

```
CMCTL>
SET LOG_DIRECTORY /disk1/user_cman_test/oracle/network/admin

CMAN_user.us.example.com parameter log_directory set to
/disk1/user_cman_test/oracle/network/admin

The command completed successfully
```

2.3.16 SET LOG_LEVEL

Use the Oracle Connection Manager Control Utility `SET LOG_LEVEL` command to set Oracle Connection Manager log levels.

Purpose

To set the log level for Oracle Connection Manager.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET LOG_LEVEL [level] [-c instance_name]

CMCTL> SET LOG_LEVEL [level]
```

From Oracle Connection Manager Control utility:

Arguments

- `off`: No logging.
- `user`: User log information.
- `admin`: Administrative log information.
- `support`: Oracle Support Services log information. This is the default.

level: Specify one of the following log levels:

Usage Notes

Specify `off` to capture the minimum amount of log information. Specify `support` to capture the maximum amount.

Example

```
CMCTL> SET LOG_LEVEL SUPPORT
CMAN_user.us.example.com parameter log_level set to SUPPORT
The command completed successfully
```

REST API for SET LOG_LEVEL Command

```
POST /set/loglevel
JSON Payload
{
"log_level" : "level"
}
```

2.3.17 SET OUTBOUND_CONNECT_TIMEOUT

Use the Oracle Connection Manager Control Utility `SET OUTBOUND_CONNECT_TIMEOUT` command to specify the time limit that Oracle Connection Manager instances wait for server connections before timing out.

Purpose

To specify the maximum amount of time the Oracle Connection Manager instance waits for a valid connection with the server before timing out.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET OUTBOUND_CONNECT_TIMEOUT [time] [-c instance_name]
```

From the From the Oracle Connection Manager Control utility:

```
CMCTL> SET OUTBOUND_CONNECT_TIMEOUT [time]
```

Arguments

time: The outbound connect timeout in seconds. The default is `0`.

Example

```
CMCTL> SET OUTBOUND_CONNECT_TIMEOUT 30
CMAN_user.us.example.com parameter outbound_connect_timeout set to 30
The command completed successfully
```

REST API for SET OUTBOUND_CONNECT_TIMEOUT Command

```
POST /set/octo
JSON Payload
{
“outbound_connect_timeout” : “time”
}
```

2.3.18 SET SESSION_TIMEOUT

Use the Oracle Connection Manager Control utility `SET SESSION_TIMEOUT` command to specify the maximum duration of Oracle Connection Manager sessions.

Purpose

To specify the maximum amount of time for a session of Oracle Connection Manager.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET SESSION_TIMEOUT [time] [-c  instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET SESSION_TIMEOUT [time]
```

Arguments

time: The session timeout in seconds. The default is 0 (zero), which disables this feature.

Example

```
CMCTL> SET SESSION_TIMEOUT 60
CMAN_user.us.example.com parameter session_timeout set to 60
The command completed successfully
```

REST API for SET SESSION_TIMEOUT Command

```
POST /set/mct
JSON Payload
{
"session_timeout" : "time"
}
```

2.3.19 SET TRACE_DIRECTORY

Use the Oracle Connection Manager Control utility `SET TRACE_DIRECTORY` command to designate where Oracle Connection Manager instance trace files are written.

> **Note:**
>
> This command works only if you have not enabled Automatic Diagnostic Repository (ADR). The default is for ADR to be enabled.

Purpose

To designate where the trace files for Oracle Connection Manager instances are written.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET TRACE_DIRECTORY [directory_path] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET TRACE_DIRECTORY [directory_path]
```

Arguments

directory_path: The location of the trace directory. The default path is `ORACLE_HOME/network/trace`.

Usage Notes

Use the `SHOW PARAMETERS` command to determine the trace file locations.

Example

```
CMCTL> SET TRACE_DIRECTORY /disk1/mpurayat_newtest/oracle/network/trace
cman1 parameter trace_directory set to /disk1/mpurayat_newtest/oracle/network
/trace
The command completed successfully
```

2.3.20 SET TRACE_LEVEL

Use the Oracle Connection Manager Control utility `SET TRACE_LEVEL` command to set Oracle Connection Manager instance trace levels.

Purpose

To set the trace level for an Oracle Connection Manager instance.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SET TRACE_LEVEL [level] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SET TRACE_LEVEL [level]
```

Arguments

level: Specify one of the following log levels:

- `off`: No tracing. This is the default.
- `user`: User trace information.
- `admin`: Administrative trace information.
- `support`: Oracle Support Services trace information.

Usage Notes

Specify `off` to capture the minimum amount of trace information. Specify `support` to capture the maximum amount.

Use the `SHOW PARAMETERS` command to determine the current trace level.

Example

```
CMCTL> SET TRACE_LEVEL USER
CMAN_user.us.example.com parameter trace_level set to USER
The command completed successfully
```

REST API for SET TRACE_LEVEL Command

```
POST /set/tracelevel
JSON Payload
{
“trace_level” : “level”
}
```

2.3.21 SHOW

Use the Oracle Connection Manager Control utility `SHOW` command to display the parameters that you can use as arguments for this command.

Purpose

To display a list of parameters that you can use as arguments for the `SHOW` command. Entering one of these parameters with the command displays the parameter value or values.

Prerequisites

None

Syntax

From the operating system:

```
cmctl SHOW [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW
```

Example

```
CMCTL> SHOW
The following operations are available after show
An asterisk (*) denotes a modifier or extended command:

all             gateways        status
connections     parameters      version
defaults        rules
events          services
```

2.3.22 SHOW ALL

Use the Oracle Connection Manager Control utility `SHOW ALL` command to combine and display `SHOW PARAMETERS` and `SHOW RULES` command output.

Purpose

To combine and display output from the `SHOW PARAMETERS` and `SHOW RULES` commands.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW ALL [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW ALL
```

Example

```
CMCTL> SHOW ALL
listener_address          |
(address=(protocol=tcp)(host=users.us.example.com)(port=1630))
aso_authentication_filter |   OFF
connection_statistics     |   OFF
event_group               |   OFF
log_directory             | /disk1/user_cman_test/oracle/network/log/
log_level                 | SUPPORT
max_connections           |   256
idle_timeout              |     0
inbound_connect_timeout   |     0
session_timeout           |     0
outbound_connect_timeout  |     0
max_gateway_processes     |    16
min_gateway_processes     |     2
max_cmctl_sessions        |     4
trace_directory           | /disk1/user_cman_test/oracle/network/trace/
trace_level               |   OFF
trace_timestamp           |   OFF
trace_filelen             |     0
trace_fileno              |     0
(rule_list=
 (rule=
  (src=*)
  (dst=*)
  (srv=*)
  (act=accept)
 )
)
The command completed successfully
```

REST API for SHOW ALL Command

```
GET /show/all
```

2.3.23 SHOW CONNECTIONS

Use the Oracle Connection Manager Control utility `SHOW CONNECTIONS` command to display connection information.

Purpose

To display information about specific connections or all connections.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW CONNECTIONS [information] [in state] [gt time] [from source]
[to destination] [for service] [using gateway_process_id]
[connect_identifier_list] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW CONNECTIONS [information][in state] [gt time] [from source]
[to destination] [for service] [using gateway_process_id]
[connect_identifier_list]
```

Arguments

information: Specify one of the following values to display information about connections. Information categories include connection identifier, source, destination, service, current state, total idle time, and total elapsed time.

- `count`: The total number of connections that meet the criteria specified by the other qualifiers. This is the default.
- `detail`: All information about connections specified by the other qualifiers.

state: Specify one of the following values to specify the connection state:

- `idle`: Connections that are inactive in the established state.
- `connecting`: Connections that are in the process of connecting.
- `established`: Connections that are connected and are transferring data.
- `terminating`: Connections that are disconnecting.

If you do not specify a state, then `SHOW CONNECTIONS` defaults to all possible states. If the time qualifier is included under these conditions, then the time specified is the amount of time that has elapsed since a client initiated a connection.

> **Note:**
>
> This argument is not supported with Oracle Connection Manager in Traffic Director mode.

time: Use the following format to specify connections greater than the time indicated:

`gt[`*hh*`:`*mm*`:]`*ss*

> **Note:**
>
> This argument is not supported with Oracle Connection Manager in Traffic Director mode.

source: Specify one of the following formats to specify the source address:

- `from` *IP*
- `from` *hostname*
- `from` *subnet*

destination: Specify one of the following formats to specify the destination address:

- *to IP*
- `to` *hostname*
- `to` *subnet*

service: Use the *service_name* format to request a service:

gateway_process_id: Use the following format to specify connections that are proxied by the gateway process indicated:

```
using gateway_process_id
```

connect_identifier_list: Separate multiple connection identifiers using a space.

Usage Notes

Connections are sorted by gateway process identifier and connection identifier, in ascending order.

Issuing `SHOW CONNECTIONS` without an argument displays all connections.

Examples

The following command displays a detailed description of connections in any state. The elapsed time of the connection must be greater than 1 hour and 30 minutes. The connection source is the specified subnet, and the destination is the specified host name.

```
CMCTL> SHOW CONNECTIONS gt 1:30:00 from 192.0.2.32/24 to host1
```

The following command displays the number of connections proxied by Oracle Connection Manager using the gateway process identifier 0 that have been in the idle state more than 30 minutes:

```
CMCTL> SHOW idle CONNECTIONS count gt 30:00 using 0
```

The following command displays a detailed description of connections that are connected to the service `sales.us.example.com`:

```
CMCTL> SHOW established CONNECTIONS detail for sales.us.example.com
```

REST API for SHOW CONNECTIONS Command

```
POST /show/connections
JSON Payload
{
“count” : “[yes | no]”,
“in”: “[ idle | connecting | established | terminated]”,
“gt” : “ time elapsed since client connection, [hh:mm:]ss format”,
“from” : “[ IP | hostname | subnet]”,
```

```
"to" : "[ IP | hostname | subnet]",
"for" : "service name",
"using" : " gateway process id",
"connect_ids" : [id1, id2]
}
```

An example to show the established connection details for `sales.us.example.com` using the `json` schema is:

```
{
"count": "no",
"state" : "established",
"for" : "sales.us.example.com"
}
```

Additional Statistics Shown in Traffic Director Mode

Each connection to Oracle Connection Manager in Traffic Director mode displays these additional statistics:

- `Source Host Name`: Host name of the client connection.
- `Source Process Id`: Process Id of the connected client.
- `Source Program Name`: The name of the connected client program.
- `Destination Hostname`: Host name of the database server to which the client is connected through Oracle Connection Manager.
- `State`: State of the inbound connection with one of the following values
  - `THREAD WAIT`: Connection is waiting for a worker thread, not seen in dedicated threads mode
  - `ACTIVE`: Connection is transferring data, occupying the thread
  - `IDLE`: Connection is established but inactive, can still occupy the thread if `tdm_bind_thread=true` in `cman.ora`
- `Idle time`: Cumulative time in`µs` the connection is in`IDLE` state.
- `Thread Wait time`: Cumulative time in`µs` the connection is in `THREAD WAIT` state. It is always `0` in dedicated threads mode.
- `Active time`: Cumulative time in µs the connection is in `ACTIVE` state.
- `PRCP State`: State of the inbound connection with respect to the Proxy Resident Connection Pool (`PRCP`) and can be one of the following values
  - `WAIT`: Connection is waiting for a session from the `PRCP`
  - `CHECKED-OUT`: Connection is holding an outbound session from `PRCP` but not making any OCI calls
  - `ACTIVE`: Connection is holding an outbound session from `PRCP` and busy with OCI calls
  - `CHECKED-IN`: Connection released the `CHECKED-OUT` session back to the `PRCP`
  - `NO STATE`: Clients to a service without a configured `PRCP` configured have this state

- `PRCP Wait time`, `PRCP Checked-out time`, and `PRCP Active time`: Cumulative time in`µs` the connection is in `PRCP WAIT`, `CHECKED-OUT`, and `ACTIVE` states. All these three states are zero in case of non-`PRCP` service.
- `Total Session Gets`: Total count of`PRCP` session get requests from this connection. It is always `1` if `PRCP` is not configured.
- `Session Get Hits`: Number of times a session is found existing in the `PRCP` out of all the requests. It is always `0` if `PRCP` is not configured.

2.3.24 SHOW DEFAULTS

Use the Oracle Connection Manager Control utility `SHOW DEFAULTS` command to display default parameter settings.

Purpose

To display default parameter settings.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW DEFAULTS [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW DEFAULTS
```

Example

```
CMCTL> SHOW DEFAULTS
listener_address          |
(address=(protocol=tcp)(host=users.us.example.com)(port=1521))
aso_authentication_filter |   OFF
connection_statistics     |   OFF
event_group               |   OFF
log_directory             | /disk1/user_cman_test/oracle/network/log/
log_level                 | SUPPORT
max_connections           |   256
idle_timeout              |     0
inbound_connect_timeout   |     0
session_timeout           |     0
outbound_connect_timeout  |     0
max_gateway_processes     |    16
min_gateway_processes     |     2
max_cmctl_sessions        |     4
trace_directory           | /disk1/user_cman_test/oracle/network/trace/
trace_level               |   OFF
trace_timestamp           |   OFF
trace_filelen             |     0
trace_fileno              |     0
The command completed successfully
```

REST API for SHOW DEFAULTS Command

```
GET /show/defaults
```

2.3.25 SHOW EVENTS

Use the Oracle Connection Manager Control utility `SHOW EVENTS` command to display events that are currently operating.

Purpose

To display the events that are in operation.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW EVENTS [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW EVENTS
```

Example

```
CMCTL> SHOW EVENTS
Event Groups:
memory_ops
The command completed successfully
```

2.3.26 SHOW GATEWAYS

Use the Oracle Connection Manager Control utility `SHOW GATEWAYS` command to display the statuses of gateway processes.

Purpose

To display the current status of a specific gateway process or processes. Statistics displayed include number of active connections, number of peak active connections, total number of connections handled, and number of connections refused.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW GATEWAYS [gateway] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW GATEWAYS [gateway]
```

Arguments

gateway: The identifier of the gateway or gateways whose status to display.

Issuing `SHOW GATEWAYS` without an argument displays the status of all gateway processes.

Usage Notes

To display multiple gateways, use a space to separate the identifiers when entering the command.

Example

```
CMCTL> SHOW GATEWAYS 1
Gateway ID                    1
Gateway state                 READY
Number of active connections  0
Peak active connections       0
Total connections             0
Total connections refused     0
The command completed successfully
```

REST API for SHOW GATEWAYS Command

```
POST /show/gateways
JSON Payload
{
“gateway_ids” : [id1, id2.. ]
}
```

2.3.27 SHOW PARAMETERS

Use the Oracle Connection Manager Control utility `SHOW PARAMETERS` command to display the parameter settings for an instance.

Purpose

To display current parameter settings for an instance.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW PARAMETERS [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW PARAMETERS
```

Usage Notes

Several configuration parameters can be dynamically modified using the `SET` command. Therefore, the information that `SHOW PARAMETERS` displays might be different from what appears in the `cman.ora` file.

Example

```
CMCTL> SHOW PARAMETERS
listener_address           |
(address=(protocol=tcp)(host=users.us.example.com)(port=1630))
aso_authentication_filter |    ON
connection_statistics     |    ON
event_group               | (memory_ops)
log_directory             | /disk1/user_cman_test/oracle/network/log/
log_level                 | SUPPORT
max_connections           |   256
idle_timeout              |     0
inbound_connect_timeout   |     0
session_timeout           |     0
outbound_connect_timeout  |     0
max_gateway_processes     |    16
min_gateway_processes     |     2
max_cmctl_sessions        |     4
trace_directory           | /disk1/user_cman_test/oracle/network/trace/
trace_level               | SUPPORT
trace_timestamp           |   OFF
trace_filelen             |     0
trace_fileno              |     0
The command completed successfully
```

REST API for SHOW PARAMETERS Command

```
GET /show/parameters
```

2.3.28 SHOW RULES

Use the Oracle Connection Manager Control Utility `SHOW RULES` command to display an instance access control list.

Purpose

To display the access control list currently used by the instance.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW RULES [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW RULES
```

```
cmctl SHOW RULES [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW RULES
```

Usage Notes

You can update the rules list by issuing the `RELOAD` command.

Example

```
CMCTL> SHOW RULES
Number of filtering rules currently in effect: 5
(rule_list=
  (rule=
    (src=usunnae12)
    (dst=usunnae13)
    (srv=*)
    (act=accept)
    (action_list=(mit=120)(mct=1800)(conn_stats=on)(aut=off))
  )
  (rule=
    (src=usunnae12)
    (dst=usunnae14)
    (srv=service2)
    (act=accept)
  )
  (rule=
    (src=*)
    (dst=usunnae15)
    (srv=*)
    (act=accept)
    (action_list=(mit=120)(mct=3000)(moct=200)(aut=on))
  )

  (rule=
    (src=*)
    (dst=usunnae16)
    (srv=*)
    (act=reject)
    (action_list=(moct=20)(aut=on))
  )

  (rule=
    (src=users.us.example.com)
    (dst=users.us.example.com)
    (srv=cmon)
    (act=accept)
    (action_list=(mit=100)(mct=1130)(moct=200)(aut=on))
  )
)
```

REST API for SHOW RULES Command

```
GET /show/rules
```

2.3.29 SHOW SERVICES

Use the Oracle Connection Manager Control utility `SHOW SERVICES` command to display Oracle Connection Manager instance information.

Purpose

To display comprehensive information about Oracle Connection Manager instances. The information displayed includes the number of handlers for the gateway and CMADMIN processes, listening ports of handlers, and the number of connections, both refused and current.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW SERVICES [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW SERVICES
```

Example

```
CMCTL> SHOW SERVICES
Services Summary...
Proxy service "cmgw" has 1 instance(s).
  Instance "cman", status READY, has 2 handler(s) for this service...
    Handler(s):
      "cmgw001" established:0 refused:0 current:0 max:256 state:ready
         <machine: user-sun, pid: 29190>
         (ADDRESS=(PROTOCOL=tcp)(HOST=user-sun)(PORT=33175))
      "cmgw000" established:0 refused:0 current:0 max:256 state:ready
         <machine: user-sun, pid: 29188>
         (ADDRESS=(PROTOCOL=tcp)(HOST=user-sun)(PORT=33174))
Service "cmon" has 1 instance(s).
  Instance "cman", status READY, has 1 handler(s) for this service...
    Handler(s):
      "cmon" established:0 refused:0 current:0 max:4 state:ready
         <machine: user-sun, pid: 29184>
         (ADDRESS=(PROTOCOL=tcp)(HOST=users)(PORT=33168))
The command completed successfully
```

REST API for SHOW SERVICES Command

```
GET /show/services
```

2.3.30 SHOW STATS

Use the Oracle Connection Manager Control utility `SHOW STATS` command with the `-REG` option to display statistics about database service registration commands.

Purpose

To display statistics about the number of registration commands that the Oracle Connection Manager (CMAN) listener receives while handling client connection requests.

Using these statistics, you can monitor service registration or service update operations, such as `REGISTER`, `UPDATE`, `RE-REGISTER`, and `UN-REGISTER`. This also helps you evaluate the traffic and overhead of these operations at CMAN.

Prerequisites

CMAN must be running.

Syntax

From the operating system:

```
cmctl SHOW STATS -REG
```

From the Oracle Connection Manager Control (`CMCTL`) utility:

```
CMCTL> SHOW STATS -REG
```

Usage Notes

- In addition to `CMCTL`, you can run this command from the Listener Control (`LSNRCTL`) utility.
- If you enter `SHOW STATS -REG` without any argument, then the output displays a global-level data for all instances, registered services, handlers allocated to each service, listening endpoints, and access control lists (ACLs).
- Recent count is a periodic count of all commands received by the listener from the last reset, that is, from the time you cleared the `Recent` section using `-clear`. If you have not used `-clear` till now, then this field displays a cumulative count (total number of commands collected since the CMAN instance started).

  For a detailed description about other sections of the output, see *Oracle Database Net Services Administrator's Guide*.

Arguments and Examples

Argument and Description	Example

To display a recent and cumulative count of all registration commands at a global level:
`-reg`
	`CMCTL:cman1> show stats -reg`
	`------------------------`
	`Global Level:`
	`Recent`
	`Recent Duration: 5 days 17 hr. 15 min. 25 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 2 2 2 4 0`
	`Updates 3 0 0 12 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 3 3 3 6 0`
	`Updates 4 0 0 12 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	`The command completed successfully`

To display a recent and cumulative count of all registration commands for the specified instance name:
`-inst` *`instance_name`*
	`CMCTL:cman1> show stats -reg -inst sales1`
	`------------------------`
	`Instance Name: sales1`
	`Recent`
	`Recent Duration: 5 days 15 hr. 17 min. 18 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 2 2 2 4 0`
	`Updates 3 0 0 12 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 2 2 2 4 0`
	`Updates 3 3 0 12 0`
	`Re-Register 0 0 3 0 0`
	`Un-Register 0 0 0 0 0`
	`The command completed successfully`

Argument and Description	Example

To display a recent and cumulative count of registration commands for all instances:
`-all_inst`
	`CMCTL:cman1> show stats -reg -all_inst`
	`------------------------`
	`Instance Name: sales1`
	`Recent`
	`Recent Duration: 5 days 15 hr. 3 min. 2 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 1 1 1 2 0`
	`Updates 21 0 0 5 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 1 1 1 4 0`
	`Updates 25 0 2 5 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	`------------------------`
	`Instance Name: sales2`
	`Recent`
	`Recent Duration: 2 days 5 hr. 3 min. 2 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 1 1 1 2 0`
	`Updates 2 0 0 3 0`
	`Re-Register 0 1 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 1 1 1 3 0`
	`Updates 10 0 2 5 0`
	`Re-Register 0 3 0 0 0`
	`Un-Register 0 0 0 0 0`
	
	`The command completed successfully`

To display a recent and cumulative count of all registration commands for the specified service name:
`-serv` *`service_name`*
	`CMCTL:cman1> show stats -reg -serv sales.us.example.com`
	`------------------------`
	`Service Name: sales.us.example.com`
	`Recent Duration: 5 days 15 hr. 4 min. 10 sec`
	`Flags Goodness Delta`
	`Recent 2 0 0`
	`Cumulative 0 0 0`
	`The command completed successfully`

To display a recent and cumulative count of registration commands for all database services:
`-all_serv`
	`CMCTL:cman1> show stats -reg -all_serv`
	`------------------------`
	`Service Name: sales.us.example.com`
	`Recent Duration: 5 days 15 hr. 5 min. 25 sec`
	`Flags Goodness Delta`
	`Recent 1 2 0`
	`Cumulative 2 0 0`
	`------------------------`
	`Service Name: employee.us.example.com`
	`Recent Duration: 8 days 2 hr. 5 min. 25 sec`
	`Flags Goodness Delta`
	`Recent 1 0 0`
	`Cumulative 1 0 0`
	`The command completed successfully`

Argument and Description	Example

To reset the `Recent` section to `0` after fetching the values:
`-clear`
This gives you a fresh set of data in the `Recent` section. You can use `-clear` with all the `SHOW STATS` arguments.
	`CMCTL:cman1> show stats -reg -clear`
	`------------------------`
	`Global Level:`
	`Recent`
	`Recent Duration: 0 days 0 hr. 0 min. 4 sec`
	`Command Instance Service ENDP Handler INF`
	`Registration 0 0 0 0 0`
	`Updates 0 0 0 0 0`
	`Re-Register 0 0 0 0 0`
	`Un-Register 0 0 0 0 0`
	`Cumulative`
	`Registration 3 3 3 6 0`
	`Updates 4 0 0 13 0`
	`Re-Register 1 0 0 0 0`
	`Un-Register 0 1 0 0 0`
	`The command completed successfully`

Related Topics

- *Oracle Database Net Services Administrator's Guide*

2.3.31 SHOW STATUS

Use the Oracle Connection Manager Control utility `SHOW STATUS` command to display Oracle Connection Manager instance information.

Purpose

To display basic information about the instance, including version, start time, and current statistics.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHOW STATUS
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW STATUS
```

Example

```
CMCTL> SHOW STATUS
Status of the Instance
----------------------
Instance name             CMAN_user.us.example.com
Version                   CMAN for Linux: Version 23.4.0.0.0
Start date                12-MAR-2024 14:50:35
Uptime                    0 days 1 hr. 25 min. 24 sec
Num of gateways started   2
Average Load level        0
```

```
Log Level                 SUPPORT
Trace Level               OFF
Instance Config file      /disk1/user_cman_test/oracle/network/admin/cman.ora
Instance Log directory    /disk1/user_cman_test/oracle/network/log/
Instance Trace directory  /disk1/user_cman_test/oracle/network/trace/
The command completed successfully
```

REST API for SHOW STATUS Command

```
GET /show/status
```

2.3.32 SHOW VERSION

Use the Oracle Connection Manager Control utility `SHOW VERSION` command

Purpose

To display the current version and name of the Oracle Connection Manager Control utility.

Prerequisites

None

Syntax

From the operating system:

```
cmctl SHOW VERSION [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHOW VERSION
```

Examples

```
CMCTL> SHOW VERSION
CMAN for Linux: Version 23.4.0.0.0
The command completed successfully
```

REST API for SHOW VERSION Command

```
GET /show/version
```

2.3.33 SHUTDOWN

Use the Oracle Connection Manager Control utility `SHUTDOWN` command to shut down gateway processes or an entire Oracle Connection Manager instance.

Purpose

To shut down specific gateway processes or the entire Oracle Connection Manager instance.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SHUTDOWN [gateways gateway] [normal | abort] [timeout value] [notify]
[-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SHUTDOWN [gateways gateway] {normal | abort} [timeout value] [notify]
```

Arguments

`gateways`: To shut down a specific gateway. To specify more than one gateway, separate gateways using a space.

`normal`: To reject new connections and terminate after existing connections close. This is the default.

`abort`: To shut down Oracle Connection Manager immediately, and close all open connections.

`timeout value`: To specify the amount of time in seconds that Oracle Connection Manager must wait before shutting down.

`notify`: To notify the client that Oracle Connection Manager is shutting down.

Usage Notes

Running the `SHUTDOWN` command without an argument shuts down all of the gateways.

Examples

```
CMCTL> SHUTDOWN GATEWAYS 0

CMCTL> SHUTDOWN ABORT

CMCTL> SHUTDOWN NOTIFY
```

2.3.34 STARTUP

Use the Oracle Connection Manager Control utility `STARTUP` command to start Oracle Connection Manager.

Purpose

To start Oracle Connection Manager.

Prerequisites

Another Oracle Connection Manager instance configured with the same protocol address must not be running.

Syntax

From the operating system:

```
cmctl STARTUP [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> STARTUP
```

Usage Notes

- Before running this command, you must use the `ADMINISTER` command to select an instance to start.

  Issuing this command starts all instance components, which are the listener, `CMADMIN`, and the gateway processes. The command fails if any one of these components is already running.
- The use of password access to Oracle Connection Manager parameters is desupported in Oracle Database 23ai.
  Oracle provides an enhanced connection method, Local Operating System Authentication" (LOSA), which permits only the user who started CMAN to perform admin operations. This method is consistent with other operating system authentication methods used with Oracle Database. If you are currently using password access to CMAN, then Oracle recommends that you remove the CMAN password, and instead rely on LOSA.

Example

```
CMCTL> STARTUP
Starting Oracle Connection Manager instance cman_1. Please wait...
CMAN for Linux: Version 23.4.0.0.0
Status of the Instance
----------------------
Instance name             cman_1
Version                   CMAN for Linux: Version 23.4.0.0.0
Start date                22-Feb-2024 01:16:55
Uptime                    0 days 0 hr. 0 min. 9 sec
Num of gateways started   8
Average Load level        0
Log Level                 SUPPORT
Trace Level               OFF
Instance Config file      $ORACLE_HOME/network/admin/cman.ora
Instance Log directory    $ORACLE_BASE/diag/netcman/node_name/cman_1/alert
Instance Trace directory  $ORACLE_BASE/diag/netcman/node_name/cman_1/trace
The command completed successfully
```

STARTUP -MIGRATE

Use the `STARTUP -MIGRATE` parameter to start Oracle Connection Manager (CMAN) in migration mode. Youcan start a new instance of CMAN in migration mode, and migrate connected sessions from the already running instance of Oracle CMAN.

Prerequisites

A CMAN instance with the same configuration as the new instance must be running in a different *`ORACLE_HOME`* on the same host where the new CMAN is being started.

Syntax

From the operating system:

```
cmctl STARTUP -MIGRATE [-c instance_name]
```

Usage Notes

This command starts new instance components, such as, the listener, `CMADMIN`, and the gateway processes.

The new listener inherits the listening endpoints and the listen queue from the old listener. It also accepts new connection requests.

The old gateway processes migrate the connected sessions to new gateways. This migration happens without the client or the server intervention.

The old listener exits after processing the pending connections. The old instance, `CMADMIN`, and the gateway processes will exit as soon as migration is complete or after 7 minutes timeout.

Use `instance_name_old` as the instance name to monitor the old instance.

Example

```
CMCTL STARTUP -MIGRATE -C cman_1
CMCTL for Linux: Version 23.4.0.0.0

Copyright (c) 1996, 2024, Oracle.  All rights reserved.

Current instance cman_1_ is already started
Connecting to (DESCRIPTION=(address=(protocol=tcp)(host=localhost)
(port=2556)))
CMAN Session Migration Stats
----------------------------
No of Gateways               |     1
Total Connections            |     0
TCP Connections              |     0
TCPS Connections(Migratable) |     0

Starting CMAN Session Migration....
-----------------------------------
Old CMADMIN address alias parameter cman_1_old set to
(configuration=(ADDRESS=(PROTOCOL=ipc)(KEY="#124470.1")
(KEYPATH=/var/tmp/.oracle_754500)))
.
Starting Oracle Connection Manager instance cman_1. Please wait...
CMAN for Linux: Version 23.4.0.0.0 - Development
Status of the Instance
----------------------
Instance name             cman_1
Version                   CMAN for Linux: Version 23.4.0.0.0
Start date                27-FEB-2024 05:31:03
Uptime                    0 days 0 hr. 0 min. 9 sec
Num of gateways started   1
```

```
Average Load level        0
Log Level                 SUPPORT
Trace Level               SUPPORT
Instance Config file      /network/admin/cman.ora
Instance Log directory $ORACLE_BASE/diag/netcman/node_name/cman_1/alert
Instance Trace directory $ORACLE_BASE/diag/netcman/node_name/cman_1/
trace
The command completed successfully.
Now session migration will be initiated by gateways separately....
```

2.3.35 SUSPEND GATEWAY

Use the Oracle Connection Manager Control utility `SUSPEND GATEWAY` command to specify the gateway processes than cannot accept new client connections.

Purpose

To specify which gateway processes will no longer accept new client connections.

Prerequisites

Oracle Connection Manager must be running.

Syntax

From the operating system:

```
cmctl SUSPEND GATEWAY [gateway_process_id] [-c instance_name]
```

From the Oracle Connection Manager Control utility:

```
CMCTL> SUSPEND GATEWAY [gateway_process_id]
```

Arguments

gateway_process_id: The gateway process that will no longer accept new connections. Specify multiple gateway processes by entering a space between entries.

Issuing `SUSPEND GATEWAY` without an argument suspends all gateway processes.

Usage Notes

Use the `RESUME GATEWAYS` command to enable gateway processes to accept new connections.

Example

```
CMCTL> SUSPEND GATEWAY 1
The command completed successfully
```

REST API for SUSPEND GATEWAY Command

```
POST /suspend
{
"gateway" : "gateway id"
}
```

3
Syntax Rules for Configuration Files

Learn the syntax rules for configuring Oracle Net Services parameters, keywords, addresses, and naming methods.

- Overview of Configuration File Syntax
  Create Oracle Net Services configuration files using syntax rules and standard conventions.
- Syntax Rules for Configuration Files
  Follow the structure, hierarchy, and character requirements for configuration files.
- Network Character Set for Keywords
  Use the permitted character set for keyword values and network character sets.
- Permitted Listener and Net Service Name Character Set
  Create listener names and net service names for clients that comply with Oracle Net Services character set requirements.

3.1 Overview of Configuration File Syntax

Create Oracle Net Services configuration files using syntax rules and standard conventions.

The Oracle Net Services configuration files contain parameters that include keyword-value pairs. Keyword-value pairs are surrounded by parentheses:

```
parameter=(keyword=value)
```

Some keywords have other keyword-value pairs as their values:

```
(keyword=
    (keyword1=value1)
    (keyword2=value2))
```

For example, the address portion of the local naming configuration file (`tnsnames.ora`) can include lines such as:

```
(ADDRESS=
   (PROTOCOL=tcp)
   (HOST=sales-server)
   (PORT=1521))
```

Set up your configuration files using indentation to show what keyword is the parent or owner of other keyword-value pairs. If you do not indent your files this way, then you must still indent

a wrapped line by at least one space, or it will be misread as a new parameter. The following syntax is acceptable:

```
(ADDRESS=(PROTOCOL=tcp)
  (HOST=sales-server)(PORT=1521))
```

The following syntax is not acceptable:

```
(ADDRESS=(PROTOCOL=tcp)
(HOST=sales-server)(PORT=1521))
```

3.2 Syntax Rules for Configuration Files

Follow the structure, hierarchy, and character requirements for configuration files.

The following rules apply to the configuration file syntax:

- Any keyword in a configuration file that begins a parameter that also includes one or more keyword-value pairs must be in the far-left column of a line. If you indent the keyword by one or more spaces, then Oracle interprets the indented keyword as a continuation of the previous line.
- All characters must be part of the network character set.
- Keywords are not case-sensitive. However, values can be case-sensitive depending on your operating system and protocol.
- In keyword-value pairs, spaces around the equal sign (=) are optional.
- There is a hierarchy of keywords that requires that some keywords are always followed by others. At any level in the hierarchy, keywords can be listed in any order. For example, the following entries are equally valid:

  ```
  (ADDRESS=
     (PROTOCOL=TCP)
     (HOST=sales-server)
     (PORT=1521))

  (ADDRESS=
     (PROTOCOL=tcp)
     (PORT=1521)
     (HOST=sales-server))
  ```

- Keywords cannot contain spaces.
- Values must not contain spaces, unless the values with spaces are enclosed within double quotation marks (") or single quotation marks (').
- If the keyword-value pair consists of a single word, or a concatenation of words on either side of the equal sign, then no parentheses are needed.
- The maximum length of a connect descriptor is 4KB.
- You can include comments by using the number sign (#) at the beginning of a line. Anything following the number sign to the end of the line is considered a comment.

3.3 Network Character Set for Keywords

Use the permitted character set for keyword values and network character sets.

The network character set for keyword values consists of the following characters. Connect descriptors must be made up of single-byte characters.

```
A-Z, a-z
0-9
( ) < > / \
, . : ; ' "=- _
$ + * # & ! % ? @
```

Within this character set, the following symbols are reserved:

```
( ) = \ " ' #
```

Reserved symbols are used as delimiters, not as part of a keyword or a value, unless the keyword or value has quotation marks. If you have a value that contains reserved symbols, then use either single or double quotation marks to enclose the value. To include quotation marks within a value that is surrounded by quotation marks, use different quotation marks. The backslash (\) is used as an escape character.

You can use the following characters within a connect descriptor, but not in a keyword or value:

- Space
- Tab
- Carriage return
- Newline

3.4 Permitted Listener and Net Service Name Character Set

Create listener names and net service names for clients that comply with Oracle Net Services character set requirements.

Listener names and net service names are limited to the following character set:

```
[a...z] [A...Z] [0...9] _
```

The first character in the listener name or net service name must be an alphanumeric character. In general, names up to 64 characters are acceptable. In addition, a database service name must match the global database name that is defined by the database administrator, which consists of a database name and the database domain. Both net service names and global database names are not case-sensitive.

4
Protocol Address Configuration

Learn how to configure connections for Oracle Database instances and clients.

A network object is identified by a protocol address. When a connection is made, the client and the receiver of the request (a listener or Oracle Connection Manager) are configured with identical protocol addresses. The client uses this address to send the connection request to a particular network object location. The recipient "listens" for requests on this address, and grants connections based on its address information matching the client's information.

- Protocol Addresses
  The protocol address comprises `ADDRESS` and `ADDRESS_LIST` elements.
- Protocol Parameters
  The listener and Oracle Connection Manager are identified by protocol addresses.
- Recommended Port Numbers
  Oracle recommends that you use the default port numbers for client and Oracle Connection Manager connections.
- Port Number Limitations
  Use this procedure to configure listeners to use a system port number in the 1 to 1024 range.

4.1 Protocol Addresses

The protocol address comprises `ADDRESS` and `ADDRESS_LIST` elements.

- ADDRESS
  The `ADDRESS` networking parameter specifies the protocol address under the `ADDRESS_LIST` or `DESCRIPTION` parameter.
- ADDRESS_LIST
  The `ADDRESS_LIST` networking parameter specifies the number of protocol addresses sharing common characteristics.

4.1.1 ADDRESS

The `ADDRESS` networking parameter specifies the protocol address under the `ADDRESS_LIST` or `DESCRIPTION` parameter.

Purpose

To define a protocol address.

Usage Notes

Put this parameter under an `ADDRESS_LIST` or `DESCRIPTION` parameter. A `DESCRIPTION` is used in a `tnsnames.ora` or a `listener.ora` file.

Example

```
(ADDRESS=
 (PROTOCOL=tcp)
 (HOST=sales-server)
 (PORT=1521))
```

Related Topics

- Protocol Parameters
  The listener and Oracle Connection Manager are identified by protocol addresses.
- *Oracle Database Global Data Services Concepts and Administration Guide*

4.1.2 ADDRESS_LIST

The `ADDRESS_LIST` networking parameter specifies the number of protocol addresses sharing common characteristics.

Purpose

To define a list of protocol addresses that share common characteristics.

Usage Notes

This parameter is not mandatory when specifying multiple addresses.

Example

```
(ADDRESS_LIST=
  (LOAD_BALANCE=on)
  (ADDRESS=
   (PROTOCOL=tcp)
   (HOST=sales-server)
   (PORT=1521))
  (ADDRESS=
   (PROTOCOL=tcp)
   (HOST=hr-server)
   (PORT=1521)))
```

4.2 Protocol Parameters

The listener and Oracle Connection Manager are identified by protocol addresses.

The following table lists the parameters that Oracle protocol support uses:

Table 4-1 Protocol-Specific Parameters

Protocol	Parameter	Description
IPC	`PROTOCOL`	Specify `ipc` as the value.

Table 4-1 (Cont.) Protocol-Specific Parameters

Protocol	Parameter	Description
IPC	KEYPATH	On UNIX variants, the IPC protocol uses the UNIX domain socket and this socket creates an internal file for client/server communication. The parameter keypath specifies the location where this file is created. If you use keypath, then use the same value of a version greater than 18 on the client and listener sides.
IPC	KEY	Specify a unique name for the service. Oracle recommends using the service name or the Oracle system identifier (SID) of the service. Example: (PROTOCOL=ipc)(KEY=sales)
Named Pipes	PROTOCOL	Specify nmp as the value.
Named Pipes	SERVER	Specify the Oracle server name.
Named Pipes	PIPE	Specify the pipe name used to connect to the database server. This is the same PIPE keyword specified on the server with Named Pipes. This name can be any name. Example: (PROTOCOL=nmp)(SERVER=sales)(PIPE=dbpipe0)
SDP	PROTOCOL	Specify sdp as the value.
SDP	HOST	Specify the host name or IP address of the computer.
SDP	PORT	Specify the listening port number. Example: (PROTOCOL=sdp)(HOST=sales-server)(PORT=1521) (PROTOCOL=sdp)(HOST=192.0.2.204)(PORT=1521)
TCP/IP	PROTOCOL	Specify tcp as the value.
TCP/IP	HOST	Specify the host name or IP address of the computer.
TCP/IP	PORT	Specify the listening port number. Example: (PROTOCOL=tcp)(HOST=sales-server)(PORT=1521) (PROTOCOL=tcp)(HOST=192.0.2.204)(PORT=1521)
TCP/IP with TLS	PROTOCOL	Specify tcps as the value.
TCP/IP with TLS	HOST	Specify the host name or IP address of the computer.
TCP/IP with TLS	PORT	Specify the listening port number. Example: (PROTOCOL=tcps)(HOST=sales-server)(PORT=2484) (PROTOCOL=tcps)(HOST=192.0.2.204)(PORT=2484)
Exadirect	PROTOCOL	Specify exadirect as the value.
Exadirect	HOST	Specify the IP address of the InfiBand interface.

Table 4-1 (Cont.) Protocol-Specific Parameters

Protocol	Parameter	Description
Exadirect	PORT	Specify the listening port number. Example: (PROTOCOL=exadirect)(HOST=sales-server)(PORT=2484) (PROTOCOL=tcps)(HOST=192.0.2.204)(PORT=1522)
Websocket	PROTOCOL	Specify ws as the value; use this protocol only as a web server back-end database server.
Websocket	HOST	Specify the host name or IP address of the computer.
Websocket	PORT	Specify the listening port number. Example: (protocol=ws)(host=sales-server)(port=1524)
Secure Websocket	PROTOCOL	Specify ws as the value; use this protocol on the client side to connect to a web server with websocket protocol support. Configure the web server to make a websocket connection to the database listener. Configure the wallet in sqlnet.ora. Use SQLNET.URI for mapping on web server.
Secure Websocket	HOST	Specify the host name or IP address of the web server with websocket support.
Secure Websocket	PORT	Specify the listening port number. Example: (protocol=wss)(host=sales-server)(port=1524)

4.3 Recommended Port Numbers

Oracle recommends that you use the default port numbers for client and Oracle Connection Manager connections.

Table 4-2 Recommended Port Numbers

Port	Description
1521	Default listening port for client connections to the listener. This port number can change to the officially registered port number of 2483 for TCP/IP, and 2484 for TCP/IP with TLS.
1521	Default and officially registered listening port for client connections to Oracle Connection Manager.
1830	Default and officially registered listening port for administrative commands to Oracle Connection Manager.

4.4 Port Number Limitations

Use this procedure to configure listeners to use a system port number in the 1 to 1024 range.

Oracle accepts port numbers from 1 to 65535. However, port numbers below 1024 are typically reserved. Only privileged processes can listen for TCP connections on ports below 1024.

To configure a listener to listen on a port number lower than 1024, complete the following procedure:

> **Note:**
>
> This procedure is a guideline. Your operating system can require a different procedure.

1. Use Oracle Net Configuration Assistant or Oracle Net Manager to configure the listener with protocol addresses and other configuration parameters.
2. Log in as the `root` user on the machine that has the listener.
3. Set file ownership and access permissions for the listener executable (`tnslsnr`) and the dependent shared libraries, so that these files can be modified only by the `root` user.
4. Starting with the `root` directory, ensure that the permissions of the individual directories in the path names to these files share the same ownership and access permissions.
5. Start the listener as the `root` user.
6. Enter the following command at the prompt:

   ```
   tnslsnr listener_name -user user -group group
   ```

   In the preceding command, the following options are used:

Table 4-3 tnslsnr Utility Options

Options	Description
listener_name	Specify the name of the listener to configure. If omitted, then the default name `LISTENER` is used.
user	Specify the user whose privileges you want the listener to use when super user (`root`) privileges are not needed. After performing the privileged operations, the listener gives up `root` privileges irreversibly.
group	Specify the group whose privileges you want the listener to use when super user (`root`) group privileges are not needed. After performing the privileged operations, the listener gives up `root` group privileges irreversibly.

   During this step, the listener changes from `root` to the user and group privileges that you specify. All operations are done with the specified user and group privileges, except for the issuing of the system calls that are needed to listen on configured endpoints. The listener reverts to the `root` user to listen on reserved addresses, such as TCP port numbers that are lower than 1024.

   After the listener starts listening on all of its endpoints that you configured in the `listener.ora` file, it permanently switches to the specified user and group. At that point, the listener gives up the `root` privilege that it initially had. The `-user` and `-group`

command line arguments only accept user and group identifiers specified in numeric form.

For example, to run a listener called `mylsnr` with `root` privileges, and to have it use privileges of the Oracle user with the user identifier (UID) of `37555`, and with OSDBA group `dba` membership, with a group identifier (GID) of 16, enter the following command at the prompt:

```
tnslsnr mylsnr -user 37555 -group 16
```

7. After the listener starts, you can administer it with Listener Control utility.

> **Caution:**
>
> - Oracle recommends that the user under whose privileges the listener process runs is the `oracle` user, or a similarly privileged user with whose privileges the listener process normally runs on the operating system.
> - Do not leave the listener process running as the `root` user. Running processes as the super user is a security risk.

5
Parameters for sqlnet.ora Files

This chapter describes the `sqlnet.ora` file parameters.

- Overview of Profile Configuration Files
  Learn about profile configuration files.
- Profile Parameters in sqlnet.ora Files
  These are the `sqlnet.ora` profile configuration parameters that you use to administer database clients and servers.
- ADR Diagnostic Parameters in sqlnet.ora
  Diagnostic data for critical errors is stored in the `sqlnet.ora` Automatic Diagnostic Repository (ADR).
- Non-ADR Diagnostic Parameters in sqlnet.ora Files
  Learn about `sqlnet.ora` parameters that you use when you disable ADR.

5.1 Overview of Profile Configuration Files

Learn about profile configuration files.

The `sqlnet.ora` file is the Net Services profile configuration file. The `sqlnet.ora` file resides on clients and databases. You store and implement profiles using this file. You can also configure the database with access control parameters in the `sqlnet.ora` file. These parameters specify whether clients are allowed or denied access to a database based on the parameter settings.

The `sqlnet.ora` file enables you to:

- Specify the client domain to append to unqualified names
- Prioritize naming methods
- Enable logging and tracing features
- Route connections through specific processes
- Configure parameters for external naming
- Configure Oracle Advanced Security
- Use protocol-specific parameters to restrict access to the database

Oracle Net searches for the `sqlnet.ora` file in the following locations and in the following order:

- In the directory specified in the `TNS_ADMIN` environment variable, if it is set.
- In the *`ORACLE_BASE_HOME`*`/network/admin` directory.
- In the *`ORACLE_HOME`*`/network/admin` directory.

Note:

- The settings in the `sqlnet.ora` file apply to all pluggable databases (PDBs) in multitenant container database environments.
- Oracle Net Services supports the IFILE parameter in the `sqlnet.ora` file, with up to three levels of nesting. The parameter is added manually to the file. The following is an example of the syntax:

  ```
  IFILE=/tmp/listener_em.ora
  IFILE=/tmp/listener_cust1.ora
  IFILE=/tmp/listener_cust2.ora
  ```

  Refer to *Oracle Database Reference* for additional information.
- With Oracle Instant Client, the `sqlnet.ora` file is located in the subdirectory of the Oracle Instant Client software. For example, in the `/opt/oracle/instantclient_release_number/network/admin` directory.
- In the read-only Oracle home mode, the `sqlnet.ora` file default location is *`ORACLE_BASE_HOME`*`/network/admin`.
- In the read-only Oracle home mode, the parameters are stored in the `ORACLE_BASE_HOME` location by default.

5.2 Profile Parameters in sqlnet.ora Files

These are the `sqlnet.ora` profile configuration parameters that you use to administer database clients and servers.

Note:

Starting with Oracle Database 23ai, the parameter `ENCRYPTION_WALLET_LOCATION` is desupported.

To store and retrieve the TDE wallet, use the `WALLET_ROOT` structure (introduced with Oracle Database 18c).

The `WALLET_ROOT` parameter is described in *Oracle Database Advanced Security Guide*.

- ACCEPT_MD5_CERTS
  The `sqlnet.ora` profile parameter `ACCEPT_MD5_CERTS` accepts MD5 signed certificates.
- ACCEPT_SHA1_CERTS
  Use the `sqlnet.ora` profile parameter `ACCEPT_SHA1_CERTS` to determine whether SQL Net accepts SHA1 signed certificates.

- ALLOWED_WEAK_CERT_ALGORITHMS
  Use the `sqlnet.ora` parameter `ALLOWED_WEAK_CERT_ALGORITHMS` to allow the use of deprecated certification algorithms as an exception.
- AZURE_DB_APP_ID_URI
  Use the `AZURE_DB_APP_ID_URI` parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).
- CLIENT_CERTIFICATE
  Use the `CLIENT_CERTIFICATE` parameter to specify the file system path to a client certificate that authenticates the database client.
- CLIENT_ID
  Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.
- EXADIRECT_FLOW_CONTROL
  The `sqlnet.ora` profile parameter `EXADIRECT_FLOW_CONTROL` enables or disables Exadirect flow control.
- EXADIRECT_RECVPOLL
  Use the `sqlnet.ora` parameter `EXADIRECT_RECVPOLL` to specify the amount of time that a receiver polls for incoming data.
- DEFAULT_SDU_SIZE
  Use the `sqlnet.ora` profile parameter to specify the session data unit size (SDU) for connections.
- DISABLE_INTERRUPT
  Use the `sqlnet.ora` profile parameter `DISABLE_INTERRUPT` to disable Oracle Net handling of a `SIGINIT` signal in client applications.
- DISABLE_OOB
  Use the `sqlnet.ora` profile parameter `DISABLE_OOB` to enable or disable Oracle Net to send or receive out-of-band break messages using urgent data from the underlying protocol.
- DISABLE_OOB_AUTO
  Use the `sqlnet.ora` profile parameter `DISABLE_OOB_AUTO` to disable server path checks for out-of-band break messages at the time of the connection.
- IPC.KEYPATH
  Use the `sqlnet.ora` profile parameter `IPC.KEYPATH` to specify the destination directory where the internal file is created for UNIX domain sockets.
- KERBEROS5_PRINCIPAL
  Use the `KERBEROS5_PRINCIPAL` parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.
- NAMES.DEFAULT_DOMAIN
  Use the `sqlnet.ora` profile parameter `NAMES.DEFAULT_DOMAIN` to set the name of the domain in which clients most often look up names resolution requests.
- NAMES.DIRECTORY_PATH
  Use the `sqlnet` parameter `NAMES.DIRECTORY_PATH` to specify the order of the naming methods for client name resolution lookups.
- NAMES.LDAP_AUTHENTICATE_BIND
  Use the `sqlnet` parameter `NAMES.LDAP_AUTHENTICATE_BIND` to specify whether the LDAP naming adapter should authenticate using a specified wallet when it connects to the LDAP directory to resolve connect string names.

- NAMES.LDAP_AUTHENTICATE_BIND_METHOD
  Use the `sqlnet` parameter `NAMES.LDAP_AUTHENTICATE_BIND_METHOD` to specify an authentication method for the client LDAP naming adapter.
- NAMES.LDAP_CONN_TIMEOUT
  Use the `sqlnet` parameter `NAMES.LDAP_CONN_TIMEOUT` to specify the number of seconds that indicates that a non-blocking connect timeout to the LDAP server occurred.
- NAMES.LDAP_PERSISTENT_SESSION
  Use the `sqlnet` parameter `NAMES.LDAP_PERSISTENT_SESSION` to specify whether the LDAP naming adapter should leave the session with the LDAP server open after name lookups are complete.
- NAMES.NIS.META_MAP
  Use the `sqlnet` parameter `NAMES.NIS.META_MAP` to specify the map file to use to map Network Information Service (NIS) attributes to an NIS mapname.
- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.
- OCI_CONFIG_FILE
  Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.
- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.
- OCI_IAM_URL
  Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.
- OCI_TENANCY
  Use the `OCI_TENANCY` parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.
- PASSWORD_AUTH
- RECV_BUF_SIZE
  Use the `sqlnet` parameter `RECV_BUF_SIZE` to specify buffer space limit for session receive operations.
- REDIRECT_URI
  Use the `REDIRECT_URI` parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.
- SDP.PF_INET_SDP
  Use the `sqlnet` parameter `SDP.PF_INET_SDP` to specify the protocol family or address family constant for the SDP protocol on your system.
- SEC_USER_AUDIT_ACTION_BANNER
  Use the `sqlnet` parameter `SEC_USER_AUDIT_ACTION_BANNER` to specify a text file that contains the banner contents that warn users about user action auditing.

- SEC_USER_UNAUTHORIZED_ACCESS_BANNER
  Use the `sqlnet` parameter `SEC_USER_UNAUTHORIZED_ACCESS_BANNER` to specify the file that contains the banner contents that warn users about unauthorized database access.
- SEND_BUF_SIZE
  Use the `sqlnet` parameter `SEND_BUF_SIZE` to specify the buffer space limit for session send operations.
- SQLNET.ALLOW_WEAK_CRYPTO
  Use the `sqlnet.ora` compatibility parameter `SQLNET.ALLOW_WEAK_CRYPTO` to configure your client-side network connection by reviewing the specified encryption and crypto-checksum algorithms.
- SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS
  Use the `sqlnet.ora` compatibility parameter `SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS` to configure your server-side network connection by reviewing the specified encryption and crypto-checksum algorithms.
- SQLNET.ALLOWED_LOGON_VERSION_CLIENT
  Use the `sqlnet` parameter `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` to define minimum authentication protocols that servers acting as clients to other servers can use for connecting to Oracle Database instances.
- SQLNET.ALLOWED_LOGON_VERSION_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to set the minimum authentication protocol that is permitted when connecting to Oracle Database instances.
- SQLNET.AUTHENTICATION_SERVICES
  Use the `sqlnet.ora` parameter `SQLNET.AUTHENTICATION_SERVICES` to enable one or more authentication services.
- SQLNET.CLIENT_REGISTRATION
  Use the `sqlnet.ora` parameter `SQLNET.CLIENT_REGISTRATION` to set a unique identifier for the client computer.
- SQLNET.CLOUD_USER
  Use the `sqlnet.ora` parameter `SQLNET.CLOUD_USER` to specify a user name for web server `HTTP` basic authentication.
- SQLNET.COMPRESSION
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION` to enable or disable data compression.
- SQLNET.COMPRESSION_ACCELERATION
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_ACCELERATION` to specify the use of hardware accelerated version of compression using this parameter if it is available for that platform.
- SQLNET.COMPRESSION_LEVELS
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_LEVELS` to specify the compression level.
- SQLNET.COMPRESSION_THRESHOLD
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_THRESHOLD` to specify the minimum data size for which compression is needed.
- SQLNET.CRYPTO_CHECKSUM_CLIENT
  Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_CLIENT` to specify the desired data integrity behavior when this client or server acting as a client connects to a server.

- SQLNET.CRYPTO_CHECKSUM_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_SERVER` to specify the data integrity behavior when a client or another server acting as a client connects to this server.
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT
  Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` to specify a list of data integrity algorithms that this client or server acting as a client uses.
- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` to specify the data integrity algorithms that this server or client to another server uses, in order of intended use.
- SQLNET.DBFW_PUBLIC_KEY
  Use the `sqlnet.ora` parameter `SQLNET.DBFW_PUBLIC_KEY` to provide Oracle Database Firewall public keys to the Advanced Security Option (ASO) by specifying the file that stores the public keys.
- SQLNET.DOWN_HOSTS_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.DOWN_HOSTS_TIMEOUT` to specify the amount of time in seconds that server hosts down state information remains in the client cache.
- SQLNET.ENCRYPTION_CLIENT
  Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_CLIENT` to set the encryption behavior when this client or server acting as a client connects to a server.
- SQLNET.ENCRYPTION_SERVER
  The `sqlnet.ora` parameter `SQLNET.ENCRYPTION_SERVER` specifies the encryption behavior when a client or a server acting as a client connects to this server.
- SQLNET.ENCRYPTION_TYPES_CLIENT
  Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_TYPES_CLIENT` to specify the encryption algorithms this client or the server acting as a client uses.
- SQLNET.ENCRYPTION_TYPES_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_TYPES_SERVER` to specify the encryption algorithms this server uses in the order of the intended use.
- SQLNET.EXPIRE_TIME
  Use the `sqlnet.ora` parameter `SQLNET.EXPIRE_TIME` to specify how often, in minutes, to verify that client and server connections are active.
- SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS
  Use the `sqlnet.ora` parameter `SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS` to ignore the value that is set for the parameter `SQLNET.ENCRYPTION_SERVER` for TCPS connections. This disables ANO encryption on the TCPS listener.
- SQLNET.INBOUND_CONNECT_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.INBOUND_CONNECT_TIMEOUT` to specify the amount of time that clients have to connect with the database and authenticate.
- SQLNET.FALLBACK_AUTHENTICATION
  Use the `sqlnet.ora` parameter `SQLNET.FALLBACK_AUTHENTICATION` to specify whether to attempt password-based authentication if Kerberos authentication fails.
- SQLNET.KERBEROS5_CC_NAME
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.

- SQLNET.KERBEROS5_CLOCKSKEW
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CLOCKSKEW` to specify how much time elapses before a Kerberos credential is considered out-of-date.
- SQLNET.KERBEROS5_CONF
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CONF` to specify the path name to the Kerberos configuration file that contains the realm for the default Key Distribution Center (KDC) and that maps realms to KDC hosts.
- SQLNET.KERBEROS5_CONF_LOCATION
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CONF_LOCATION` to specify the directory for the Kerberos configuration file. The `SQLNET.KERBEROS5_CONF_LOCATION` parameter also specifies that the file is created by the system and not by the client.
- SQLNET.KERBEROS5_KEYTAB
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_KEYTAB` to specify the path name to the Kerberos principal or, secret, key mapping file that extracts keys and decrypts incoming authentication information.
- SQLNET.KERBEROS5_REALMS
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_REALMS` to specify the complete path name to the Kerberos realm translation file that maps a host name or domain name to a realm.
- SQLNET.KERBEROS5_REPLAY_CACHE
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_REPLAY_CACHE` to specify that the replay cache is stored in operating system-managed memory on the server, and that file-based replay cache is not used.
- SQLNET.OUTBOUND_CONNECT_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.OUTBOUND_CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.
- SQLNET.RADIUS_ALLOW_WEAK_CLIENTS
  Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.
- SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` to allow weak Oracle Database clients to use RADIUS authentication.
- SQLNET.RADIUS_ALTERNATE
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE` to specify an alternate RADIUS server to be used when the primary server is unavailable.
- SQLNET.RADIUS_ALTERNATE_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_PORT` to specify the listening port of an alternate RADIUS server.
- SQLNET.RADIUS_ALTERNATE_RETRIES
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_RETRIES` to specify the number of times that the database resends messages to alternate RADIUS servers.
- SQLNET.RADIUS_ALTERNATE_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TIMEOUT` to set the time for an alternate RADIUS server to wait for a response.

- SQLNET.RADIUS_ALTERNATE_TLS_HOST
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_HOST` to specify the host name of an alternate RADIUS server to be used when the primary server is unavailable.
- SQLNET.RADIUS_ALTERNATE_TLS_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_PORT` to specify the listening port of an alternate RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION` to specify the location of a primary RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION_INTERFACE
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_INTERFACE` to specify the class that contains the user interface for interacting with users.
- SQLNET.RADIUS_AUTHENTICATION_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_PORT` to specify the listening port of a primary RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION_RETRIES
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_RETRIES` to specify the number of times the database should resend messages to a primary RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TIMEOUT` to specify the amount of time that the database should wait for a response from a primary RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION_TLS_HOST
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_HOST` to specify the host name of a primary RADIUS server.
- SQLNET.RADIUS_AUTHENTICATION_TLS_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_PORT` to specify the listening port of a primary RADIUS server.
- SQLNET.RADIUS_CHALLENGE_KEYWORD
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CHALLENGE_KEYWORD` to set the keyword for requesting a challenge from the RADIUS server.
- SQLNET.RADIUS_CHALLENGE_RESPONSE
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CHALLENGE_RESPONSE` to enable or disable challenge responses.
- SQLNET.RADIUS_CLASSPATH
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CLASSPATH` to set the path for Java classes and JDK Java libraries.
- SQLNET.RADIUS_SECRET
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_SECRET` to specify the location of a RADIUS secret key.
- SQLNET.RADIUS_SEND_ACCOUNTING
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_SEND_ACCOUNTING` to enable and disable accounting.

- SQLNET.RADIUS_TRANSPORT_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_TRANSPORT_PROTOCOL` to control the transport protocol that the Oracle Database server must use for communicating with the RADIUS server.
- SQLNET.RECV_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.RECV_TIMEOUT` to specify the duration of time that a database client or server should wait for data from a peer after establishing a connection.
- SQLNET.SEND_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.SEND_TIMEOUT` to specify the duration of time in which a database must complete send operations to clients after establishing connections.
- SQLNET.URI
  Use the `sqlnet.ora` parameter `SQLNET.URI` to specify a database client URI mapping on a web server.
- SQLNET.USE_HTTPS_PROXY
  Use the `sqlnet.ora` parameter `SQLNET.USE_HTTPS_PROXY` to enable forward HTTP proxy tunneling for client connections.
- SQLNET.WALLET_OVERRIDE
  Use the `sqlnet.ora` parameter `SQLNET.WALLET_OVERRIDE` to determine whether a client should override strong authentication credentials with the password credential from the stored wallet.
- SSL_ALLOW_WEAK_DN_MATCH
  Use the `sqlnet.ora` parameter `SSL_ALLOW_WEAK_DN_MATCH` to allow the earlier weaker distinguished name (DN) matching behavior during server-side certificate validation.
- SSL_CERTIFICATE_ALIAS
  Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_ALIAS` to specify the alias of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.
- SSL_CERTIFICATE_THUMBPRINT
  Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_THUMBPRINT` to specify the thumbprint of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.
- SSL_CERT_REVOCATION
  Use the `sqlnet.ora` parameter `SSL_CERT_REVOCATION` to configure revocation checks for certificates.
- SSL_CRL_FILE
  Use the `sqlnet.ora` parameter `SSL_CRL_FILE` to specify the name of the file in which you assemble the certificate revocation list (CRL) for client authentication.
- SSL_CRL_PATH
  Use the `sqlnet.ora` parameter `SSL_CRL_PATH` to specify the destination directory of the certificate revocation list (CRL) for client authentication.
- SSL_CIPHER_SUITES
  Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).
- SSL_CLIENT_AUTHENTICATION
  Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).

- SSL_ENABLE_WEAK_CIPHERS
  Use the `sqlnet.ora` parameter `SSL_ENABLE_WEAK_CIPHERS` to enable the use of weak Transport Layer Security (TLS) cipher suites.
- SSL_EXTENDED_KEY_USAGE
  Use the `sqlnet.ora` parameter `SSL_EXTENDED_KEY_USAGE` to specify the purpose certificate keys.
- SSL_SERVER_DN_MATCH
  Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.
- SSL_VERSION
  Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.
- TCP.ALLOWED_PROXIES
  Use the `sqlnet.ora` parameter `TCP.ALLOWED_PROXIES` to specify a list of the Oracle Connection Manager (CMAN) addresses that can forward client IP address to the database server.
- TCP.CONNECT_TIMEOUT
  Use the `sqlnet.ora` parameter `TCP.CONNECT_TIMEOUT` to specify the amount of time in which a client must establish TCP connections to database servers.
- TCP.EXCLUDED_NODES
  Use the `sqlnet.ora` parameter `TCP.EXCLUDED_NODES` to specify which clients are denied access to the database.
- TCP.INVITED_NODES
  Use the `sqlnet.ora` parameter `TCP.INVITED_NODES` to specify which clients are allowed access to the database.
- TCP.NODELAY
  Use the `sqlnet.ora` parameter `TCP.NODELAY` to preempt delays in buffer flushing within the TCP/IP protocol stack.
- TCP.QUEUESIZE
  Use the sqlnet.ora parameter TCP.QUEUESIZE to configure the maximum length of queues for pending connections on TCP listening sockets.
- TCP.VALIDNODE_CHECKING
  Use the `sqlnet.ora` parameter `TCP.VALIDNODE_CHECKING` to enable and disable valid node checking for incoming connections.
- TENANT_ID
  Use the `TENANT_ID` parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.
- TNSPING.TRACE_DIRECTORY
  Use the `sqlnet.ora` parameter `TNSPING.TRACE_DIRECTORY` to specify the destination directory for the TNSPING utility trace file, `tnsping.trc`.
- TNSPING.TRACE_LEVEL
  Use the `sqlnet.ora` parameter `TNSPING.TRACE_LEVEL` to enable or disable TNSPING utility tracing at a specified level.
- TOKEN_AUTH
- TOKEN_LOCATION
  Use the `TOKEN_LOCATION` parameter to specify the directory location where token file is stored for token-based authentication.

- USE_CMAN
  Use the `sqlnet.ora` parameter `USE_CMAN` to specify client routing to Oracle Connection Manager.
- USE_DEDICATED_SERVER
  Use the sqlnet.ora parameter USE_DEDICATED_SERVER to append `(SERVER=dedicated)` to the `CONNECT_DATA` section of the connect descriptor that the client uses.
- WALLET_LOCATION
  Use the `WALLET_LOCATION` parameter to specify the location of Oracle wallets.
- BEQUEATH_DETACH
  Use the `sqlnet.ora` parameter to enable and disable handling signals on Linux and UNIX systems.

5.2.1 ACCEPT_MD5_CERTS

The `sqlnet.ora` profile parameter `ACCEPT_MD5_CERTS` accepts MD5 signed certificates.

Purpose

To enable `sqlnet` to accept MD5 signed certificates. In addition to `sqlnet.ora`, you must also set this parameter in `listener.ora`.

Default

`FALSE`

Values

- `TRUE` to accept MD5 signed certificates
- `FALSE` to not accept MD5 signed certficates

5.2.2 ACCEPT_SHA1_CERTS

Use the `sqlnet.ora` profile parameter `ACCEPT_SHA1_CERTS` to determine whether SQL Net accepts SHA1 signed certificates.

Purpose

To determine whether `sqlnet` accepts SHA1 signed certificates. In addition to setting this parameter in `sqlnet.ora`, you must also set this parameter in `listener.ora`.

The use of SHA-1 with `DBMS_CRYPTO`, `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated.

Using SHA-1 (Secure Hash Algorithm 1) with the parameters `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated in this release, and can be desupported in a future release. Using SHA-1 ciphers with `DBMS_CRYPTO` is also deprecated (`HASH_SH1`, `HMAC_SH1`). Instead of using SHA1, Oracle recommends that you start using a stronger SHA-2 cipher in place of the SHA-1 cipher.

Default

TRUE

Values

- `TRUE` to accept SHA1 signed certificates
- `FALSE` to not accept SHA1 signed certificates

5.2.3 ALLOWED_WEAK_CERT_ALGORITHMS

Use the `sqlnet.ora` parameter `ALLOWED_WEAK_CERT_ALGORITHMS` to allow the use of deprecated certification algorithms as an exception.

Purpose

To allow the use of earlier weaker algorithms for backward compatibility. This is useful for environments that still require the use of certificates associated with deprecated algorithms, such as `MD5` or `SHA1` signed certificates.

Usage Notes

Starting in Oracle Database 23ai, the `ALLOW_MD5_CERTS` and `ALLOW_SHA1_CERTS` `sqlnet.ora` parameters are deprecated.

Instead of these parameters, use the `ALLOWED_WEAK_CERT_ALGORITHMS` `sqlnet.ora` parameter, which is new with Oracle Database 23ai.

If `ALLOWED_WEAK_CERT_ALGORITHMS` is set, then Oracle Database ignores `ALLOW_MD5_CERTS` and `ALLOW_SHA1_CERTS`. If `ALLOWED_WEAK_CERT_ALGORITHMS` is not set, then Oracle Database checks and uses the `ALLOW_MD5_CERTS` and `ALLOW_SHA1_CERTS` settings.

Values

`MD5 | SHA1`

Oracle Database allows you to use only those weak algorithms that you set here:

- When set to `MD5`, it allows `MD5` but disables `SHA1`.
- When set to `SHA1`, it allows `SHA1` but disables `MD5`.
- When set to `MD5,SHA1`, it allows both `MD5` and `SHA1`.

Ensure that you enclose the values in parenthesis. If you want to specify both `MD5` and `SHA1`, then separate the values with a comma.

Default

`SHA1`

Examples

```
ALLOWED_WEAK_CERT_ALGORITHMS=(SHA1)


ALLOWED_WEAK_CERT_ALGORITHMS=(MD5,SHA1)
```

Related Topics

- *Oracle Database Security Guide*

5.2.4 AZURE_DB_APP_ID_URI

Use the `AZURE_DB_APP_ID_URI` parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).

Purpose

To specify the unique app ID URI of the database instance registered with Azure AD. This is the protected resource identifier (on Azure AD) for which the database client application requests an access token during token-based authentication.

Usage Notes

- You must set this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` authentication flows.
- This URI value is used to compose the authorization scope (permission) of your database token request:

  ```
  $Scope = "database_app_id_uri/scope"
  ```

  For example:

  ```
  $Scope = "https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3/session:scope:connect"
  ```

  In this example, `https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3` is the app ID URI and `session:scope:connect` is the scope.
- For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the app ID URI value by logging in to the Azure portal. This is listed as the Application ID URI value on the App registrations - Overview page of the Azure Active Directory service.

Specify the app ID URI in the following format:

`Azure_AD_tenancy_url/application_(client)_id`

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-
a12b-aa00123b2cd3
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In these examples, the optional `CLIENT_ID`, `TENANT_ID`, and `REDIRECT_URI` parameters are not specified. Thus, the client automatically gets these values from the Azure SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

5.2.5 CLIENT_CERTIFICATE

Use the `CLIENT_CERTIFICATE` parameter to specify the file system path to a client certificate that authenticates the database client.

Purpose

File system path to a client certificate that authenticates the database client application registered with Microsoft Azure Active Directory (Azure AD). A client certificate is the digital certificate of an Azure cloud resource, and the client uses this certificate as a credential to prove its identity when requesting an Azure AD access token. This is used for the `AZURE_SERVICE_PRINCIPAL` token-based authentication flow.

Usage Notes

This is an optional parameter. When a client secret is not configured, the client driver reads the file system path of a client certificate from the `AZURE_CLIENT_CERTIFICATE_PATH` environment variable in the Azure SDK configuration. You can use this parameter along with the `TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL` setting to override the default certificate path. Note that this parameter is ignored if the client driver is configured with a client secret.

If you have not configured the SDKs, then you must set this parameter (along with other required parameters, such as `CLIENT_ID` and `TENANT_ID`). Otherwise, an error message appears prompting you to configure all required parameters.

For all supported clients (JDBC-thin clients, ODP.NET Core classes, and ODP.NET Managed Driver classes), you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

Full path (including a file name) to the Azure certificate file

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (CLIENT_CERTIFICATE=ORACLE_HOME/.azure/certificates/my-app.pem))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&CLIENT_CERTIFICATE=ORACLE_HOME/.azure/certificates/my-app.pem
```

In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*

- TOKEN_AUTH

5.2.6 CLIENT_ID

Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify the ID of the Azure AD application. This is the unique application (client) ID assigned to your application by Azure AD when the application is registered. This application is your database client that requests to get an access token for the user during Azure AD token-based authentication.

Usage Notes

You use this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` authentication flows.

This is an optional parameter. You can set it in these scenarios:

- For the `AZURE_MANAGED_IDENTITY` authentication flow (applicable to client-side or server-side applications hosted on Azure environments, such as Azure App Service or Azure virtual machine), the client driver uses a system-assigned managed identity. A system-assigned managed identity is an implicit identity assigned by Azure AD to your application, and is configured in the Azure SDK by default.

  You can use this parameter to explicitly assign the client ID of a user-assigned managed identity to your application.
- For all other authentication flows, if you have configured the Azure SDKs, then the client driver automatically searches for the client ID in the SDK configuration. If you have not configured the SDKs, then you must set this parameter (along with other required parameters, such as `TENANT_ID` and `CLIENT_CERTIFICATE`). Otherwise, an error message appears prompting you to configure all required parameters.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the client ID value by logging in to the Azure portal. This is listed as the Application (client) ID value on the App registrations - Overview page.

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3
CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In these examples, the optional `TENANT_ID` and `REDIRECT_URI` parameters are not specified. The client driver automatically gets these values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

5.2.7 EXADIRECT_FLOW_CONTROL

The `sqlnet.ora` profile parameter `EXADIRECT_FLOW_CONTROL` enables or disables Exadirect flow control.

Purpose

To enable or disable Exadirect flow control.

Usage Notes

Set to `on`, the parameter enables Oracle Net to broadcast the available receive window to the sender. The sender limits the sends based on the receiver broadcast window.

Default

`off`

Example

```
EXADIRECT_FLOW_CONTROL=on
```

5.2.8 EXADIRECT_RECVPOLL

Use the `sqlnet.ora` parameter `EXADIRECT_RECVPOLL` to specify the amount of time that a receiver polls for incoming data.

Purpose

To specify the amount of time that a receiver polls for incoming data.

Usage Notes

You can set the parameter to a fixed value or set the parameter to `AUTO` to automatically tune the polling value.

Default

`0`

Example

```
EXADIRECT_RECVPOLL = 10

EXADIRECT_RECVPOLL = AUTO
```

5.2.9 DEFAULT_SDU_SIZE

Use the `sqlnet.ora` profile parameter to specify the session data unit size (SDU) for connections.

Purpose

To specify the session data unit (SDU) size, in bytes, for connections.

Usage Notes

Oracle recommends setting this parameter in both the client-side and server-side `sqlnet.ora` files to ensure that the same SDU size is used throughout a connection. When the configured values of client and database server do not match for a session, the lower of the two values is used.

You can override this parameter for a particular client connection by specifying the SDU parameter in the connect descriptor for a client.

Default

`8192 bytes (8 KB)`

Values

512 to 2097152 bytes

Example 5-1 Example

```
DEFAULT_SDU_SIZE=4096
```

5.2.10 DISABLE_INTERRUPT

Use the sqlnet.ora profile parameter DISABLE_INTERRUPT to disable Oracle Net handling of a SIGINIT signal in client applications.

Purpose

To disable Oracle Net handling of a SIGINIT signal in client applications.

Usage Notes

Oracle Net installs a signal handler to catch a SIGINT signal. By default, the action on receipt of a SIGINIT signal is to cancel the current operation. If you set this parameter to TRUE, then you can override the default behavior and ignore Oracle Net handling of SIGINT signals.

For details on installing and uninstalling your own signal handlers in addition to Oracle Net, see *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems*.

Default

FALSE

Example

```
DISABLE_INTERRUPT=TRUE
```

5.2.11 DISABLE_OOB

Use the sqlnet.ora profile parameter DISABLE_OOB to enable or disable Oracle Net to send or receive out-of-band break messages using urgent data from the underlying protocol.

Purpose

To enable or disable Oracle Net to send or receive out-of-band break messages using urgent data provided by the underlying protocol.

Usage Notes

Set to off, the parameter enables Oracle Net to send and receive break messages. Set to on, the parameter disables the ability to send and receive break messages. Once enabled, this feature applies to all protocols that the client uses.

Default

off

Example 5-2 Example

```
DISABLE_OOB=on
```

5.2.12 DISABLE_OOB_AUTO

Use the `sqlnet.ora` profile parameter `DISABLE_OOB_AUTO` to disable server path checks for out-of-band break messages at the time of the connection.

Purpose

To disable `sqlnet.ora` from checking for out-of-band (OOB) break messages in the server path at connection time.

Usage Notes

By default, the client determines if the server path supports out-of-band break messages at the time of establishing the connection. If `DISABLE_OOB_AUTO` is set to `TRUE`, then the client does not perform this check at connection time.

Default

`FALSE`

Example 5-3 Example

```
DISABLE_OOB_AUTO = TRUE
```

5.2.13 IPC.KEYPATH

Use the `sqlnet.ora` profile parameter `IPC.KEYPATH` to specify the destination directory where the internal file is created for UNIX domain sockets.

Purpose

To specify the destination directory where the internal file is created for UNIX domain sockets.

Usage Notes

This parameter applies only to Oracle Net usage of UNIX domain sockets and does not apply to other uses of UNIX domain sockets in Oracle Database, such as in Oracle Clusterware. If you use the `IPC.KEYPATH` parameter, then you should use the same value for `IPC_KEYPATH` on both the client and the listener on Oracle Database versions that are greater than Oracle Database 18c.

Default

The directory path is either `/var/tmp/.oracle` for Oracle Linux, Oracle Solaris or `/tmp/.oracle` for other UNIX variants.

Example

```
ipc.keypath=/home/oracleuser.
```

5.2.14 KERBEROS5_PRINCIPAL

Use the KERBEROS5_PRINCIPAL parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.

Purpose

When you configure Kerberos authentication for an Oracle Database client, you can specify multiple Kerberos principals with a single Oracle Database client.

This is an optional parameter. When specified, it is used to verify if the principal name in the credential cache (specified using KERBEROS5_CC_NAME) matches the parameter value.

Usage Notes

Use this parameter in the SECURITY section of the tnsnames.ora file, or set it in the sqlnet.ora file. Alternatively, you can set KERBEROS5_PRINCIPAL in the connect string along with the KERBEROS5_CC_NAME parameter to connect as a different Kerberos principal.

The parameter value specified in the connect string takes precedence over the value specified in the sqlnet.ora or tnsnames.ora file.

Each Kerberos principal must have a valid credential cache. Oracle Database checks KERBEROS5_PRINCIPAL against the value that is retrieved from the credential cache. If the two values do not match, then the user is not authenticated.

Examples

- For a user krbuser1, who is externally authenticated using the Kerberos principal krbprinc1@example.com and the credential cache for this principal is located at /tmp/krbuser1/krb.cc, the connect descriptor in the tnsnames.ora file is:

```
net_service_name=
(DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-svr)(PORT=1521))
      (CONNECT_DATA=(SERVICE_NAME=sales.example.com))
      (SECURITY=
          (KERBEROS5_CC_NAME=/tmp/krbuser1/krb.cc)
          (KERBEROS5_PRINCIPAL=krbprinc1@example.com)))
```

  In the sqlnet.ora file:

```
SQLNET.KERBEROS5_CC_NAME=/tmp/krbuser1/krb.cc
KERBEROS5_PRINCIPAL=krbprinc1@example.com
```

- For a user krbuser2, who is externally authenticated using the Kerberos principal krbprinc2@example.com and the credential cache for this principal is located at /tmp/krbuser2/krb.cc, the connect descriptor in the tnsnames.ora file is:

```
net_service_name=
(DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-svr)(PORT=1521))
```

```
(CONNECT_DATA=(SERVICE_NAME=sales.example.com))
(SECURITY=
    (KERBEROS5_CC_NAME=/tmp/krbuser2/krb.cc)
    (KERBEROS5_PRINCIPAL=krbprinc2@example.com)))
```

In `sqlnet.ora` file:

```
SQLNET.KERBEROS5_CC_NAME=/tmp/krbuser2/krb.cc
KERBEROS5_PRINCIPAL=krbprinc2@example.com
```

> **Note:**
>
> The connection fails if the principal in the `/tmp/krbuser1/krb.cc` file does not contain the `krbprinc1@example.com` value.

Related Topics

- SQLNET.KERBEROS5_CC_NAME
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- KERBEROS5_CC_NAME
  Use the `tnsnames.ora` parameter `KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- *Oracle Database Security Guide*

5.2.15 NAMES.DEFAULT_DOMAIN

Use the `sqlnet.ora` profile parameter `NAMES.DEFAULT_DOMAIN` to set the name of the domain in which clients most often look up names resolution requests.

Purpose

To set the domain from which the client most often looks up names resolution requests.

Usage Notes

When you set `NAMES.DEFAULT_DOMAIN`, the default domain name is automatically appended to any unqualified net service name or service name.

For example, if you set the default domain to `www.example.com`, then Oracle searches the connect string `CONNECT scott@sales` as `www.example.com`. If the connect string includes the domain extension, such as `CONNECT scott@sales.www.example.com`, then the domain is not appended to the string.

Default

None

Example

```
NAMES.DEFAULT_DOMAIN=example.com
```

5.2.16 NAMES.DIRECTORY_PATH

Use the `sqlnet` parameter `NAMES.DIRECTORY_PATH` to specify the order of the naming methods for client name resolution lookups.

Purpose

To specify the order of the naming methods for client name resolution lookups.

Default

```
NAMES.DIRECTORY_PATH=(tnsnames, ldap, ezconnect)
```

Values

The following table shows the NAMES.DIRECTORY_PATH values for the naming methods.

Naming Method Value	Description
`tnsnames` (local naming method)	Set to resolve a network service name through the `tnsnames.ora` file on the client.
`ldap` (directory naming method)	Set to resolve a database service name, net service name, or network service alias through a directory server.
`ezconnect` or `hostname` (Easy Connect naming method)	Select to enable clients to use a TCP/IP connect identifier that consists of a host name and optional port and service name.
`nis` (external naming method)	Set to resolve service information through an existing Network Information Service (NIS).

Example

```
NAMES.DIRECTORY_PATH=(tnsnames)
```

5.2.17 NAMES.LDAP_AUTHENTICATE_BIND

Use the `sqlnet` parameter `NAMES.LDAP_AUTHENTICATE_BIND` to specify whether the LDAP naming adapter should authenticate using a specified wallet when it connects to the LDAP directory to resolve connect string names.

Purpose

To specify whether the LDAP naming adapter should attempt to authenticate using a specified wallet when it connects to the LDAP directory to resolve the service name in the connect string.

Usage Notes

When set to `FALSE`, the LDAP connection is established using an anonymous bind.

When set to `TRUE`, the LDAP connection is authenticated using an Oracle wallet. You must specify the wallet location using the WALLET_LOCATION parameter.

The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.

Values

`TRUE` | `FALSE`

Default

`FALSE`

Example

```
NAMES.LDAP_AUTHENTICATE_BIND=TRUE
```

5.2.18 NAMES.LDAP_AUTHENTICATE_BIND_METHOD

Use the `sqlnet` parameter `NAMES.LDAP_AUTHENTICATE_BIND_METHOD` to specify an authentication method for the client LDAP naming adapter.

Purpose

To specify the authentication method that the client LDAP naming adapter should use while connecting to the LDAP directory to resolve connect string names.

Usage Notes

The simple authentication method over LDAPS (LDAP over TLS connection) is supported.

You store the directory entry DN and password in an Oracle wallet. When the client connects to the LDAP server, it is authenticated using the credentials stored in this wallet. The wallet trust store must contain root certificates issued by the certificate authority of the LDAP server.

The LDAP naming adapter uses the `oracle.ldap.client.dn` and `oracle.ldap.client.password` entries from the wallet for authenticating to the LDAP server. If these entries are not present, then the client attempts an anonymous authentication using TLS or LDAPS.

Value

The parameter value is a string:

```
ldaps_simple_auth
```

Default

None

Example

```
NAMES.LDAP_AUTHENTICATE_BIND_METHOD=ldaps_simple_auth
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.19 NAMES.LDAP_CONN_TIMEOUT

Use the `sqlnet` parameter `NAMES.LDAP_CONN_TIMEOUT` to specify the number of seconds that indicates that a non-blocking connect timeout to the LDAP server occurred.

Purpose

The parameter value `-1` is for infinite timeout.

Default

`15` seconds

Values

Values are in seconds. The range is `-1` to the number of seconds that is acceptable for your environment. There is no upper limit.

To specify the number of seconds for a non-blocking connect timeout to the LDAP server.

Usage Notes

Example

```
names.ldap_conn_timeout = -1
```

5.2.20 NAMES.LDAP_PERSISTENT_SESSION

Use the `sqlnet` parameter `NAMES.LDAP_PERSISTENT_SESSION` to specify whether the LDAP naming adapter should leave the session with the LDAP server open after name lookups are complete.

Purpose

To specify whether the LDAP naming adapter should leave the session with the LDAP server open after a name lookup is complete.

Usage Notes

The parameter value is Boolean.

If you set the parameter to `TRUE`, then the connection to the LDAP server is left open after the name lookup is complete. The connection remains open for the duration of the process. If the connection is lost, then it is re-established as needed.

If you set the parameter to `FALSE`, then the LDAP connection is terminated as soon as the name lookup completes. Every subsequent look-up opens the connection, performs the look-up, and closes the connection. This option prevents LDAP from having a large number of clients connected to it at any one time.

Default

```
false
```

Example

```
NAMES.LDAP_PERSISTENT_SESSION=true
```

5.2.21 NAMES.NIS.META_MAP

Use the `sqlnet` parameter `NAMES.NIS.META_MAP` to specify the map file to use to map Network Information Service (NIS) attributes to an NIS mapname.

Purpose

To specify the map file to be used to map Network Information Service (NIS) attributes to an NIS mapname.

Default

```
sqlnet.maps
```

Example

```
NAMES.NIS.META_MAP=sqlnet.maps
```

5.2.22 OCI_COMPARTMENT

Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.

Purpose

To define the scope of your database token request. This value instructs the database client to initiate a token request to databases within the specified compartment only. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Usage Notes

The `OCI_COMPARTMENT` parameter is optional if you have not specified the `OCI_DATABASE` parameter. If you choose to set `OCI_DATABASE`, then you must also set `OCI_COMPARTMENT` to limit your token request to the specified database within that compartment.

If you do not set both the `OCI_COMPARTMENT` and `OCI_DATABASE` parameters, then the entire tenancy is the scope of your token request.

You can use this parameter along with the `PASSWORD_AUTH` and `TOKEN_AUTH` authentication settings:

- With the `PASSWORD_AUTH` configuration, the database client can only request an IAM database token using the IAM user name and IAM database password.
- With the `TOKEN_AUTH` configuration, the database client can request an IAM database token using the API-key, delegation token, security token, resource principal, or instance principal credentials. You can also enable the database client to directly retrieve the `db-token` with IAM Single-Sign On (SSO) credentials by

using the OCI_INTERACTIVE, OCI_API_KEY, OCI_INSTANCE_PRINCIPAL, OCI_DELEGATION_TOKEN, and OCI_RESOURCE_PRINCIPAL authentication flows.

Use this parameter under the SECURITY section of the tnsnames.ora file, sqlnet.ora file, Easy Connect syntax, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCID for the IAM compartment to allow access for the database token. You can get the OCID value for your compartment from the Compartments information page in the OCI console.

The compartment OCID uses this syntax:

OCI_COMPARTMENT=*compartment_OCID*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the tnsnames.ora file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345)
        (OCI_COMPARTMENT=ocid1.compartment..12345)
        (OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the sqlnet.ora file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
OCI_COMPARTMENT=ocid1.compartment..12345
OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

In the Easy Connect syntax:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_COMPARTMENT=ocid1.compartment..12345&OCI
_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH
- TOKEN_AUTH
- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

5.2.23 OCI_CONFIG_FILE

Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.

Purpose

To specify the directory location of the OCI configuration file. This file stores the client connection information for OCI Identity and Access Management (IAM) users as part of their profile. The SDK, CLI, and other OCI tools use this file to access the IAM user credentials during IAM token-based authentication.

Usage Notes

This is an optional parameter. If you do not set this parameter, then the database client gets the user's profile from the default configuration file located at `C:/user-profile/.oci/config`. You can use this parameter to override the default configuration file location. In this case, the database client searches for the profile in the location specified by `OCI_CONFIG_FILE`.

You can use this parameter along with the `TOKEN_AUTH` parameter for the `OCI_API_KEY` and `OCI_INTERACTIVE` authentication flows:

- When using the `OCI_INTERACTIVE` authentication flow, if this parameter is not set and the configuration file is also not present in the default location, then Oracle Database prompts the user for a region ID, presenting a list of region IDs from which the user can choose.
- When using the `OCI_API_KEY` authentication flow, if this parameter is not set and the default configuration file is also not present, then an ORA-50109 error message is returned. In this case, you must set this parameter to include the configuration file location.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

Full path (including a file name) to the OCI configuration file

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=OCI_INTERACTIVE)
          (OCI_CONFIG_FILE=/home/dbuser1/config))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=OCI_INTERACTIVE
OCI_CONFIG_FILE=/home/dbuser1/config
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_CONFIG_FILE=/home/dbuser1/config
```

In these examples, the optional `OCI_PROFILE` parameter is not specified. Thus, the client automatically gets the `DEFAULT` profile from the specified configuration file directory.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

5.2.24 OCI_DATABASE

Use the OCI_DATABASE parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

Purpose

To define the scope of your database token request. The database OCID value instructs the database client to initiate a token request to the specified database within your compartment. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Usage Notes

This is an optional parameter. You can set this parameter to limit the access to only a particular database. If you set OCI_DATABASE, then you must also provide specific compartment identifier using the OCI_COMPARTMENT parameter.

You can use this parameter along with the PASSWORD_AUTH and TOKEN_AUTH authentication settings:

- With the PASSWORD_AUTH configuration, the database client can only request an IAM database token using the IAM user name and IAM database password.
- With the TOKEN_AUTH configuration, the database client can request an IAM database token using the API-key, delegation token, security token, resource principal, or instance principal credentials. You can also enable the database client to directly retrieve the db-token with IAM Single-Sign On (SSO) credentials by using the OCI_INTERACTIVE, OCI_API_KEY, OCI_INSTANCE_PRINCIPAL, OCI_DELEGATION_TOKEN, and OCI_RESOURCE_PRINCIPAL authentication flows.

Specify this parameter under the SECURITY section of the tnsnames.ora file, sqlnet.ora file, Easy Connect syntax, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

OCID of the database that you want to access for the client connection. You can get the OCID value for your database from the Database details page in the OCI console.

The database OCID uses this syntax:

OCI_DATABASE=*database_OCID*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the tnsnames.ora file:

```
net_service_name=
  (DESCRIPTION=
```

```
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345)
        (OCI_COMPARTMENT=ocid1.compartment..12345)
        (OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
OCI_COMPARTMENT=ocid1.compartment..12345
OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

In the Easy Connect syntax:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_COMPARTMENT=ocid1.compartment..12345&OCI_DATAB
ASE=ocid1.autonomousdatabase.oc1.12345
```

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH
- TOKEN_AUTH
- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.

5.2.25 OCI_IAM_URL

Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Purpose

To specify the IAM URL for your REST API requests. The database client connects to this URL to retrieve the database token from IAM.

Usage Notes

You set the `OCI_IAM_URL` parameter along with the `PASSWORD_AUTH` and `OCI_TENANCY` parameters while configuring IAM token-based authentication (using the IAM user name and IAM database password to retrieve the database token). These parameters are mandatory.

With this configuration, the database client can only request an IAM database token using the IAM user name and IAM database password. The client cannot request an IAM database token for an API-key, delegation token, security token, resource principal, service principal, or instance principal.

You can also set the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters to specify the scope of your token request.

Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCI IAM endpoint URL that the database client must connect with to get the database token. This URL is specific to your region and uses this syntax:

```
<authentication_regional_endpoint>/v1/actions/
generateScopedAccessBearerToken
```

You can derive this value by replacing <*authentication_regional_endpoint*> with the API endpoint URL for your region. To obtain the appropriate API endpoint URL, see Identity and Access Management Data Plane API.

For example, if you want to use the URL as `https://auth.us-region-1.example.com`, then your OCI_IAM_URL value is:

```
https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
```

Examples

In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345))
```

```
    (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
```

In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH

5.2.26 OCI_PROFILE

Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

Purpose

To specify the profile name for IAM users. This profile is the client connection information stored in the OCI configuration file, and is used during IAM token-based authentication.

Usage Notes

This is an optional parameter. A profile named `DEFAULT` is already set in the configuration file. The database client gets the `DEFAULT` profile from the OCI configuration file (from either the default `C:/user-profile/.oci/config` directory location or the location specified by `OCI_CONFIG_FILE`). You can specify this parameter to override the `DEFAULT` profile set in the configuration file and assign a new profile name for the IAM user.

You use this parameter along with the `TOKEN_AUTH` parameter for the `OCI_API_KEY` and `OCI_INTERACTIVE` authentication flows.

When this parameter is not set and the profile is also not present in the configuration file, then an error message appears indicating that token-based authentication has failed due to the profile not existing.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Values

- `DEFAULT`: This means that no value is explicitly defined for a given profile, and the profile is inherited from the default profile set in the configuration file.

- *profile_name*: Specify a new configuration profile name (for example, ADMIN_USER) to override the DEFAULT profile set in the configuration file.

Default

DEFAULT

Examples

In the tnsnames.ora file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=OCI_INTERACTIVE)
          (OCI_CONFIG_FILE=/home/dbuser1/config))
          (OCI_PROFILE=ADMIN_USER))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the sqlnet.ora file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=OCI_INTERACTIVE
OCI_CONFIG_FILE=/home/dbuser1/config
OCI_PROFILE=ADMIN_USER
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_CONFIG_FILE=/home/dbuser1/
config&OCI_PROFILE=ADMIN_USER
```

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH
- OCI_CONFIG_FILE
  Use the OCI_CONFIG_FILE parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.

5.2.27 OCI_TENANCY

Use the OCI_TENANCY parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.

Purpose

To specify OCID of the user's tenancy (root compartment).

Usage Notes

You set this parameter along with the mandatory `PASSWORD_AUTH` and `OCI_IAM_URL` parameters while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

With this configuration, the database client can only request an IAM database token using the IAM user name and IAM database password. The client cannot request an IAM database token for an API-key, delegation token, security token, resource principal, service principal, or instance principal.

You can also set the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters to specify the scope of your token request. If you do not set the `OCI_COMPARTMENT` and `OCI_DATABASE` parameter values, then the entire tenancy is the scope of your token request.

Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCID of the user's tenancy. You can get the OCID value for your tenancy from the Tenancy information page in the OCI console.

The tenancy OCID uses this syntax:

`OCI_TENANCY=`*`tenancy_OCID`*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
```

```
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
```

In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH

5.2.28 PASSWORD_AUTH

Use the `PASSWORD_AUTH` parameter to configure an authentication method for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS). With this setting, client connections use the IAM user name and IAM database password for logging in users to the database.

Purpose

To configure either IAM database password verifier authentication or IAM token-based authentication, using the IAM user name and IAM database password for the access.

For password verifier authentication, the database server retrieves an IAM database password verifier from IAM. For token-based authentication, the database client requests a database token (`db-token`) from IAM.

Usage Notes

- Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.
- This setting instructs the database client to either use the existing password login process with the database server (password verifier authentication) or to get a token with the IAM user name and IAM database password (token-based authentication). This IAM database password is different from the OCI console password. An IAM user can set this password from the OCI console.

  See Create an OCI IAM password to use for Autonomous Databases User Authentication and Authorization.
- By default, this parameter is set to `PASSWORD_VERIFIER`. The `PASSWORD_AUTH=PASSWORD_VERIFIER` setting configures IAM database password verifier authentication. The database server retrieves an IAM database password verifier (an encrypted hash of password) from IAM to authenticate users.

  When an IAM user logs in with the IAM user name and IAM database password using `@connect_identifier`, the `PASSWORD_AUTH=PASSWORD_VERIFIER` setting along with `@connect_identifier` instructs the database client to follow the existing user name and password login process with the database server.

  You can use the `PASSWORD_AUTH` parameter to override the `tnsnames.ora` or `sqlnet.ora` setting by specifying a different value in the connect string.

- To configure IAM token-based authentication with the IAM user name and IAM database password, set `PASSWORD_AUTH=OCI_TOKEN`. The database client requests a database token (`db-token`) from IAM for the user to access the database.

  This `db-token` obtained by the client is a bearer token with an expiration time and scope, and does not come with a private key. These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.

  When an IAM user logs in with the IAM user name and IAM database password using `/@connect_identifier`, the `PASSWORD_AUTH=OCI_TOKEN` setting along with `/@connect_identifier` instructs the database client to get the token directly from an OCI IAM endpoint using a REST API request. If the IAM user is mapped to a database schema (exclusively or shared), then the login is completed.

  For the database client to retrieve the token from IAM, you must set additional parameters so that the database client can find the IAM endpoint along with additional meta-data. The additional parameters are `OCI_IAM_URL` and `OCI_TENANCY` along with the optional `OCI_COMPARTMENT` and `OCI_DATABASE`. These values enable the database client to make appropriate calls to the specified endpoint.

  The `OCI_IAM_URL` parameter specifies the API endpoint URL that the database client must connect with. The `OCI_TENANCY` parameter specifies the OCID (Oracle Cloud Identifier) of the user's tenancy. The optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters limit the scope of your request.

  This authentication method is more secure than using a password verifier because a password verifier is considered sensitive. Also, only the database client can retrieve the database token. Applications or tools cannot pass these types of tokens through the database client API.

> **Note:**
>
> You can also use other IAM user credentials (such as API-key, security token, resource principal, service principal, instance principal, or delegation token) to get the `db-token`. This `db-token` is a proof-of-possession (PoP) token. In this case, you use a different parameter setting (`TOKEN_AUTH=OCI_TOKEN`).
>
> Unlike the IAM database password that can only be used by the database client to retrieve the token, these credentials require an application or tool to retrieve the token. See TOKEN_AUTH.

Default

`PASSWORD_VERIFIER`

Values and Examples

Value	Example

For IAM database password verifier authentication:
`PASSWORD_AUTH=PASSWORD_VERIFIER`
Note: Use of IAM user name and IAM database password with the IAM database password verifier is the default configuration, and you do not need to set any additional parameters for the client.
However, if `PASSWORD_AUTH` is set to `OCI_TOKEN` in the client-side `sqlnet.ora` file, then the client tries to connect with OCI IAM to retrieve a database token using the IAM user name and IAM database password. In this case, you can override this setting for a particular connection using `PASSWORD_AUTH=PASSWORD_VERIFIER`.
	In the `tnsnames.ora` file:
	`net_service_name=`
	`  (DESCRIPTION =`
	`  (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`  (SECURITY=`
	`  (SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(PASSWORD_AUTH=PASSWORD_VERIFIER))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`  )`
	In the `sqlnet.ora` file:
	`PASSWORD_AUTH=PASSWORD_VERIFIER`

Value	Example

For IAM token-based authentication with the IAM user name and IAM database password:
`PASSWORD_AUTH=OCI_TOKEN`
Note: You must configure the TCPS protocol (`PROTOCOL=tcps`) and set the `SSL_SERVER_DN_MATCH` parameter to `TRUE` for token-based authentication.
	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=exampl`
	`e,CN=OracleContext")`
	`(PASSWORD_AUTH=OCI_TOKEN)`
	`(OCI_IAM_URL=https://`
	`auth.us-region-1.example.com/v1/`
	`actions/`
	`generateScopedAccessBearerToken)`
	
	`(OCI_TENANCY=ocid1.tenancy..12345)`
	`)`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.`
	`us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`PASSWORD_AUTH=OCI_TOKEN`
	`OCI_IAM_URL=https://auth.us-`
	`region-1.example.com/v1/actions/`
	`generateScopedAccessBearerToken`
	`OCI_TENANCY=ocid1.tenancy..12345`
	
	In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- OCI_IAM_URL
  Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).
- OCI_TENANCY
  Use the `OCI_TENANCY` parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.

- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.
- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

5.2.29 RECV_BUF_SIZE

Use the `sqlnet` parameter `RECV_BUF_SIZE` to specify buffer space limit for session receive operations.

Purpose

To specify the buffer space limit for receive operations of sessions.

Usage Notes

You can override this parameter for a particular client connection by specifying the RECV_BUF_SIZE parameter in the connect descriptor for a client.

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to the operating system-specific documentation for additional information about additional protocols that support this parameter.

Default

The default value for this parameter is operating system specific. The default for Linux 2.6 operating system is 87380 bytes.

Example

```
RECV_BUF_SIZE=11784
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.30 REDIRECT_URI

Use the `REDIRECT_URI` parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify the redirect URI (or reply URL) registered for the Azure AD application. This redirect URI is the client application address where the authorization server sends the authentication response after successfully authenticating the user. This is used for the `AZURE_INTERACTIVE` token-based authentication flow.

Usage Notes

This is an optional parameter. If you do not set this parameter, then the database client reads the redirect URI value from the Azure SDK configuration. You can use this parameter along with the `TOKEN_AUTH=AZURE_INTERACTIVE` setting to override the default location. The authorization server redirects the user to this location only if you have registered the redirect URI for the client application in the Azure portal.

For all supported clients (JDBC-thin clients, ODP.NET Core classes, and ODP.NET Managed Driver classes), you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the redirect URI by logging in to the Azure portal. These URIs are listed as Redirect URIs on the App registrations - Authentication page of the Azure Active Directory service.

Specify the redirect URI in the following format:

```
http://localhost
```

The URI may also include a port number as follows:

```
http://localhost:port
```

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (REDIRECT_URI=http://localhost:1575))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&REDIRECT_URI=http://localhost:1575
```

In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically retrieves the client and tenant ID values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

5.2.31 SDP.PF_INET_SDP

Use the `sqlnet` parameter `SDP.PF_INET_SDP` to specify the protocol family or address family constant for the SDP protocol on your system.

Purpose

To specify the protocol family or address family constant for the SDP protocol on your system.

Default

`27`

Values

Any positive integer

Example

```
SDP.PF_INET_SDP=30
```

5.2.32 SEC_USER_AUDIT_ACTION_BANNER

Use the `sqlnet` parameter `SEC_USER_AUDIT_ACTION_BANNER` to specify a text file that contains the banner contents that warn users about user action auditing.

Purpose

To specify a text file containing the banner contents that warn users about possible user action auditing.

Usage Notes

You must specify the complete path of the text file in the `sqlnet.ora` file on the server. Oracle Call Interface (OCI) applications can use OCI features to retrieve this banner and display it to users.

Default

None

Values

Name of the file for which the database owner has read permissions.

Example

```
SEC_USER_AUDIT_ACTION_BANNER=/opt/oracle/admin/data/auditwarning.txt
```

5.2.33 SEC_USER_UNAUTHORIZED_ACCESS_BANNER

Use the `sqlnet` parameter `SEC_USER_UNAUTHORIZED_ACCESS_BANNER` to specify the file that contains the banner contents that warn users about unauthorized database access.

Purpose

To specify the name of a text file containing the banner contents that warn users about unauthorized access to the database.

Usage Notes

You must specify the complete path of the text file in the `sqlnet.ora` file on the server. OCI applications can use OCI features to retrieve this banner and display it to users.

Default

None

Values

Name of the banner file for which the database owner has read permissions.

Example

```
SEC_USER_UNAUTHORIZED_ACCESS_BANNER=/opt/oracle/admin/data/unauthwarning.txt
```

5.2.34 SEND_BUF_SIZE

Use the `sqlnet` parameter `SEND_BUF_SIZE` to specify the buffer space limit for session send operations.

Purpose

To specify the buffer space limit for send operations of sessions.

Usage Notes

You can override this parameter for a particular client connection by specifying the SEND_BUF_SIZE parameter in the connect descriptor for a client.

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to the operating system-specific documentation for additional information about additional protocols that support this parameter.

Default

The default value for this parameter is operating system specific. The default for Linux 2.6 operating systems is 16 KB.

Example

```
SEND_BUF_SIZE=11784
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.35 SQLNET.ALLOW_WEAK_CRYPTO

Use the `sqlnet.ora` compatibility parameter `SQLNET.ALLOW_WEAK_CRYPTO` to configure your client-side network connection by reviewing the specified encryption and crypto-checksum algorithms.

Purpose

To configure your client-side network connection by reviewing the encryption and crypto-checksum algorithms enabled on the client and server. This ensures that the connection does not encounter compatibility issues and your configuration uses supported strong algorithms.

Usage Notes

- The `DES`, `DES40`, `3DES112`, `3DES168`, `RC4_40`, `RC4_56`, `RC4_128`, `RC4_256`, and `MD5` algorithms are deprecated in this release.

  As a result of this deprecation, Oracle recommends that you review your network encryption and integrity configuration to check if you have specified any of the deprecated weak algorithms.

  To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.
- If you set this parameter to `TRUE`, then you can specify deprecated algorithms for backward compatibility. This configuration allows patched clients to connect to unpatched servers, and thus such a connection is less secure.
- If you set this parameter to `FALSE`, then you can specify only supported algorithms so that clients and servers can communicate in a fully patched environment. The server enforces key fold-in for all Kerberos and JDBC thin clients. This configuration strengthens the connection between clients and servers by using strong native network encryption and integrity capabilities.

  Using this setting, if native network encryption or checksumming is enabled and a patched server or client attempts to communicate with an unpatched old client or server, then the connection fails with an error message.

Values

- `TRUE`
- `FALSE`

Default Value

`TRUE`

Recommended Value

FALSE

> **Note:**
>
> Before setting this parameter to FALSE, you must remove all deprecated algorithms listed in the server and client sqlnet.ora files.

Example

```
SQLNET.ALLOW_WEAK_CRYPTO = FALSE
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.ENCRYPTION_TYPES_CLIENT
  Use the sqlnet.ora parameter SQLNET.ENCRYPTION_TYPES_CLIENT to specify the encryption algorithms this client or the server acting as a client uses.
- SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT
  Use the sqlnet.ora parameter SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT to specify a list of data integrity algorithms that this client or server acting as a client uses.

5.2.36 SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS

Use the sqlnet.ora compatibility parameter SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS to configure your server-side network connection by reviewing the specified encryption and crypto-checksum algorithms.

Purpose

To configure your server-side network connection by reviewing the encryption and crypto-checksum algorithms enabled on the client and server. This ensures that the connection does not encounter compatibility issues and your configuration uses supported strong algorithms.

Usage Notes

- The DES, DES40, 3DES112, 3DES168, RC4_40, RC4_56, RC4_128, RC4_256, and MD5 algorithms are deprecated in this release.

  As a result of this deprecation, Oracle recommends that you review your network encryption and integrity configuration to check if you have specified any of the deprecated weak algorithms.

  To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.
- If you set this parameter to TRUE, then you can specify deprecated algorithms for backward compatibility. This configuration allows patched servers to connect to unpatched clients, and thus such a connection is less secure.

- If you set this parameter to `FALSE`, then you can specify only supported algorithms so that clients and servers can communicate in a fully patched environment. The server enforces key fold-in for all Kerberos and JDBC thin clients. This configuration strengthens the connection between clients and servers by using strong native network encryption and integrity capabilities.

  Using this setting, if native network encryption or checksumming is enabled and a patched server or client attempts to communicate with an unpatched old client or server, then the connection fails with an error message.

Values

- `TRUE`
- `FALSE`

Default Value

`TRUE`

Recommended Value

`FALSE`

> **Note:**
>
> Before setting this parameter to `FALSE`, you must remove all deprecated algorithms listed in the server and client `sqlnet.ora` files.

Example

```
SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS = FALSE
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.ENCRYPTION_TYPES_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_TYPES_SERVER` to specify the encryption algorithms this server uses in the order of the intended use.
- SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER
  Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` to specify the data integrity algorithms that this server or client to another server uses, in order of intended use.

5.2.37 SQLNET.ALLOWED_LOGON_VERSION_CLIENT

Use the `sqlnet` parameter `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` to define minimum authentication protocols that servers acting as clients to other servers can use for connecting to Oracle Database instances.

Purpose

To set the minimum authentication protocol allowed for clients when a server is acting as a client, such as connecting over a database link, when connecting to Oracle Database instances.

Usage Notes

The term `VERSION` in the parameter name refers to the version of the authentication protocol, not the version of the Oracle Database release.

If the version does not meet or exceed the value defined by this parameter, then authentication fails with an `ORA-28040: The database does not accept your client's authentication protocol; login denied` error.

The database password verifier for Oracle Database 10g, `10G` is no longer supported or available on Oracle Database 23ai. Refer to the database upgrade guide preinstallation chapters for information about how to identify the Oracle Database 10G database password verifiers, and how to update the database user to use the latest and most secure database password verifier cryptography.

Values

- `12a` for Oracle Database 12*c* Release 1 (12.1.0.2) or later (strongest protection)

  > **Note:**
  >
  > Using this setting, the clients can only authenticate using a de-optimized password version. For example, the `12C` password version.

- `12` for the critical patch updates CPUOct2012 and later Oracle Database 11*g* authentication protocols (default and stronger protection)

  > **Note:**
  >
  > Using this setting, the clients can only authenticate using a verifier that uses salt. For example, the `11G` or `12C` password versions.

Default

`12`

Example

If an Oracle Database 23ai database hosts a database link to an Oracle Database 19*g* database, then set the `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` parameter as follows for the database link connection to proceed:

```
SQLNET.ALLOWED_LOGON_VERSION_CLIENT=12
```

In this case, you cannot configure the more secure `SQLNET.ALLOWED_LOGON_VERSION_CLIENT` setting of `12a` on the 23ai server hosting the database link because the account on the Oracle Database 19*g* database might not have its password changed and thus might only have the `11G` verifier.

Related Topics

- *Oracle Database Reference*
- *Oracle Database Security Guide*
- *Oracle Database Upgrade Guide*

5.2.38 SQLNET.ALLOWED_LOGON_VERSION_SERVER

Use the `sqlnet.ora` parameter `SQLNET.ALLOWED_LOGON_VERSION_SERVER` to set the minimum authentication protocol that is permitted when connecting to Oracle Database instances.

Purpose

To set the minimum authentication protocol for connecting to Oracle Database instances.

Usage Notes

- **Authentication Protocol Versions**:

  The term `VERSION` in the parameter name refers to the version of the authentication protocol, not the Oracle Database release.

  A value that appears higher up in Table 5-1 is less compatible (in terms of the protocol that clients must understand in order to authenticate) but simultaneously more secure than a value that appears lower down. The server is also more restrictive in terms of the password version that must exist to authenticate any specific account. Whether a client can authenticate to a specific account depends on both the server's setting of its `SQLNET.ALLOWED_LOGON_VERSION_SERVER` parameter, as well as on the password versions that exist for the specified account. You can see the list of password versions in file `DBA_USERS.PASSWORD_VERSIONS`.

  If the client does not have the ability listed in the "Ability Required of the Client" column that corresponds to the row that matches the value of the `SQLNET.ALLOWED_LOGON_VERSION_SERVER` parameter in Table 5-1, then authentication fails with an `ORA-28040: The database does not accept your client's authentication protocol; login denied` or an `ORA-03134: Connections to this server version are no longer supported` error.

A setting of 12 (the default) enables only the 11G and 12C password versions. A setting of 12a enables only the 12C password version.

Note the following implications of setting the value to 12 or 12a:

- Releases of OCI clients earlier than Oracle Database 10*g* cannot authenticate to the Oracle database using password-based authentication.
- If an older client (such as Oracle Database 10*g*, which has the critical patch update CPUOct2012 and thus has the O5L_NP capability) attempts to authenticate to a more recent server and the server that it is authenticating to does not have the necessary 11G password version (which is required to authenticate to such an older client), then the client will receive an ORA-03134: Connections to this server version are no longer supported error message.

  To enable older Oracle Database 10*g* clients to authenticate when the server is using the default SQLNET.ALLOWED_LOGON_VERSION_SERVER=12 setting, ensure that the PASSWORD_VERSIONS value (found in DBA_USERS) for the account contains the value 11G (meaning that an SHA-1 verifier has been provisioned for the account). You may need to reset the password of the account if 11G does not appear in the list of password versions of the account. Resetting the password of the account automatically causes the server to provision the necessary 11G password version for the authentication of older clients (which have the O5L_NP capability).
- To take advantage of the 12C password version introduced in Oracle Database release 12.2, user passwords should be expired to encourage users to change their passwords and cause the new 12C password version to be generated for their account. By default, new passwords are treated in a case-sensitive fashion. When an account password is changed, the earlier 10G case-insensitive password version and the 11G password version are both automatically removed, and the new 12C password version is generated.
- JDBC Thin Client Support:

  In Oracle Database release 12.1.0.2 and later, if you set the sqlnet.ora parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER to 12a and you create a new account or change the password of an existing account, then only the new 12C password version is generated. The 12C password version is based on a SHA-2 (Secure Hash Algorithm) SHA-512 salted cryptographic hash deoptimized using the PBKDF2 (Password-Based Key Derivation Function 2) algorithm. When the database server is running with ALLOWED_LOGON_VERSION_SERVER set to 12a, it is running in exclusive mode. In this mode, to log in using a JDBC client, the JRE version must be at least version 8. The JDBC client enables its O7L_MR capability flag only when it is running with at least version 8 of the JRE.

  > **Note:**
  >
  > Check the PASSWORD_VERSIONS column of the DBA_USERS catalog view in *Oracle Database Reference* to see the list of password versions for any given account.

  If you set the sqlnet.ora parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER to 12, then the server runs in exclusive mode and only the 11G and 12C password versions (the SHA-1 and PBKDF2 SHA-2 based hashes of the password, respectively) are generated and allowed to be used. In such cases, fully-patched JDBC clients having

the CPUOct2012 patch can connect because these JDBC clients provide the `O5L_NP` client ability.

Older JDBC clients that do not have the CPUOct2012 containing the fix for the stealth password cracking vulnerability CVE-2012-3132, do not provide the `O5L_NP` client ability. Therefore, ensure that all of the JDBC clients are patched properly.

- **Desupport of Oracle Database 10G Password Verifier**

  The database password verifier for Oracle Database 10g, `10G` is no longer supported or available on Oracle Database 23ai. Refer to the database upgrade guide preinstallation chapters for information about how to identify the Oracle Database 10G database password verifiers, and how to update the database user to use the latest and most secure database password verifier cryptography.

  Be aware that the older client capabilities are not sufficient to authenticate with more modern servers because these servers use the default configuration of `ALLOWED_LOGON_VERSION_SERVER=12` and do not support the `10G` verifier. You should upgrade all clients to Oracle Database release 12*c* so that the `12C` password version can be used exclusively to authenticate. By default, Oracle Database release 11.2.0.3 and later clients have the `O5L_NP` ability, which enables the `11G` password version to be used exclusively. If you have an earlier Oracle Database client, then you must install the CPUOct2012 patch.

- **Client Capabilities**:

  The client must support certain abilities of the authentication protocol before the server will authenticate. If the client does not support a specified authentication ability, then the server rejects the connection with an `ORA-28040 "The database does not accept your client's authentication protocol; login denied."` error message.

  The following is the list of all client abilities. Some clients do not have all the listed abilities. Clients that are more recent have all of the capabilities of the older clients, but older clients tend to have fewer abilities than more recent clients. An ability that appears at the top in this list is more recent and secure than an ability that appears lower toward the bottom:

  – `O8L_LI`: The ability to support long identifiers (user names up to 128 bytes).

  – `O7L_MR`: The ability to perform the Oracle Database 10*g* authentication protocol using the `12C` password version. For JDBC clients, only those running on at least JRE version 8 offer the O7L_MR capability.

  – `O5L_NP`: The ability to perform Oracle Database 10*g* authentication protocols using the `11G` password version, and generating a session key encrypted for critical patch update CPUOct2012.

  – `O5L` (desupported with Oracle Database 23ai): The ability to perform the Oracle Database 10*g* authentication protocol using the `10G` password version.

  – `O4L` (desupported with Oracle Database 23ai): The ability to perform the Oracle9i database authentication protocol using the `10G` password version.

  – `O3L` (desupported with Oracle Database 23ai): The ability to perform the Oracle8i database authentication protocol using the `10G` password version.

- **Using the Gradual Database Password Rollover Feature**

When the gradual database password rollover feature is enabled for an account, the LOGON_INFO clause in the audit record enables you to see whether the user has logged in with the old password or whether an application has not yet been updated to log in using the new password.

For example:

```
(TYPE=(DATABASE));
(CLIENT ADDRESS=((PROTOCOL=ipc)(HOST=0.0.0.0)));
(LOGON_INFO=((VERIFIER=11G-OLD)
(CLIENT_CAPABILITIES=O5L_NP,O7L_MR,O8L_LI)));
```

- **Allowed Parameter Settings**:

  The following table describes the allowed settings of the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter, its effect on the generated password versions when an account is created or a password is changed, the ability flag required of the client to authenticate while the server has this setting, and whether the setting is considered to be an exclusive mode.

Table 5-1 SQLNET.ALLOWED_LOGON_VERSION_SERVER Settings

Value of the ALLOWED_LOGON_VERSION_SERVER Parameter	Generated Password Version	Ability Required of the Client	Meaning for Clients	Server Runs in Exclusive Mode
12a	12C	O7L_MR	Only Oracle Database 12*c* release 1 (12.1.0.2 or later) clients can connect to the server.	Yes because it excludes the use of 11G password version.
12	11G, 12C	O5L_NP	Oracle Database 11*g* release 2 (11.2.0.3 or later) clients can connect to the server. Older clients need the critical patch update CPUOct2012 or later, to gain the O5L_NP ability. Only older clients that have applied critical patch update CPUOct2012 or later can connect to the server.	Yes because it excludes the use of the 10G password version.

Values

- `12a` for Oracle Database 12*c* release 12.1.0.2 or later authentication protocols (strongest protection)
- `12` for Oracle Database 12*c* release 12.1 authentication protocols (default and recommended value)

> **Note:**
>
> - Starting with Oracle Database 12*c* Release 2 (12.2), the default value is 12.
> - For earlier releases, the value 12 can be used after the critical patch updates CPUOct2012 and later are applied.

Default

`12`

Example

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER=12
```

Related Topics

- Ensuring Against Password Security Threats by Using the 12C Password Version
- Managing Gradual Database Password Rollover for Applications
- *Oracle Database Upgrade Guide*

5.2.39 SQLNET.AUTHENTICATION_SERVICES

Use the `sqlnet.ora` parameter `SQLNET.AUTHENTICATION_SERVICES` to enable one or more authentication services.

Purpose

To enable one or more authentication services. If you have installed authentication, then Oracle recommends that you set `SQLNET.AUTHENTICATION_SERVICES` to either `NONE` or to one of the listed authentication methods.

Usage Notes

You can set this parameter in the `sqlnet.ora` file, `tnsnames.ora` file or directly as part of the connect string. Note that this parameter is called `AUTHENTICATION_SERVICE` in `tnsnames.ora`. The parameter value specified in the connect string takes precedence.

When using the `SQLNET.AUTHENTICATION_SERVICES` value `ALL` (the default value), the server attempts to authenticate using each of the following methods. The server falls back to the authentication methods that appear further down on the list if attempts to use the authentication methods appearing higher on the list were unsuccessful. When using local database password authentication (no external authentication), set `SQLNET.AUTHENTICATION_SERVICES=(NONE)` for better client performance.

- Authentication based on a service external to the database, such as a service on the network layer, Kerberos, or RADIUS.
- Authentication based on the operating system user's membership in an administrative operating system group. Group names are platform-specific. This authentication applies to administrative connections only.
- Authentication performed by the database.
- Authentication based on credentials stored in a directory server.

Operating system authentication enables access to the database using any user name and any password when an administrative connection is attempted, such as using the `AS SYSDBA` clause when connecting using SQL*Plus. An example of a connection is as follows.

```
sqlplus ignored_username/ignored_password AS SYSDBA
```

When the operating-system user who issued the preceding command is already a member of the appropriate administrative operating system group, then the connection is successful. This is because the user name and password are ignored by the server because Oracle checks the group membership first.

Default

`ALL`

> **Note:**
>
> When installing Oracle Database with Database Configuration Assistant (DBCA), you can set this parameter to `NTS` in the `sqlnet.ora` file.

Values

Authentication methods that are available with Oracle Net Services:

- `NONE` for no authentication methods, including Microsoft Windows native operating system authentication. When you set `SQLNET.AUTHENTICATION_SERVICES` to `NONE`, then the user can use a valid user name and password to access the database.
- `ALL` for all authentication methods.
- `BEQ` for native operating system authentication for operating systems other than Microsoft Windows.
- `KERBEROS5` for Kerberos authentication.
- `NTS` for Microsoft Windows native operating system authentication. In this case, the user must authenticate to the database with OS credentials using Windows native authentication. No external password is needed. NTS checks the group membership for an OS user. For example, if an OS user is a member of the `ORA_DBA` group, then the user can log in to the database as `SYSDBA`.

> **Note:**
>
> With the `SQLNET.AUTHENTICATION_SERVICES=NTS` setting, if you try to connect through SQL*Plus using NTS authentication and specify an external password (for example, `SQL*Plus SYSTEM/password`), then the connection fails with an `ORA-12638, 00000, "Failed to retrieve credentials for adapter_name.` error message. For regular user name and password based authentication, set the value to `NONE`.

- `RADIUS` for Remote Authentication Dial-In User Service (RADIUS) authentication.
- `TCPS` for TLS authentication.

Example

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
```

Related Topics

- Configuring Authentication
- Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora
- AUTHENTICATION_SERVICE
  Use the `tnsnames.ora` parameter `AUTHENTICATION_SERVICE` to enable one or more authentication services.

5.2.40 SQLNET.CLIENT_REGISTRATION

Use the `sqlnet.ora` parameter `SQLNET.CLIENT_REGISTRATION` to set a unique identifier for the client computer.

Purpose

To set a unique identifier for the client computer.

Usage Notes

This identifier is passed to the listener with any connection request and is included in the audit trail. The identifier can be any alphanumeric string up to 128 characters long.

Default

None

Example

```
SQLNET.CLIENT_REGISTRATION=1432
```

5.2.41 SQLNET.CLOUD_USER

Use the `sqlnet.ora` parameter `SQLNET.CLOUD_USER` to specify a user name for web server `HTTP` basic authentication.

Purpose

To specify a user name for web server `HTTP` basic authentication.

Usage Notes

When you use a secure websocket protocol, the client uses this user as the user name for authentication. The password for this user should be stored in a wallet using `mkstore` commands.

Perform the following configuration steps to use `HTTP` basic authentication with secure websockets:

1. Create a wallet using the `orapki` utility.

   ```
   orapki wallet create -wallet wallet_directory
   ```
   Example
   ```
   orapki wallet create -wallet /app/wallet
   ```

2. Add a web server public certificate.

   ```
   orapki wallet -wallet  wallet_directory  -trusted_cert -cert
   web_server_public_certificate_in_pem_format
   ```
   Example
   ```
   orapki wallet -wallet  /app/wallet  -trusted_cert -cert  server_cert.txt
   ```

3. Add the web server user name to `sqlnet.ora`. This user name is only used for authenticating the web server. This is not a database user name. After web server authentication, the web server connects to the back-end database server and database authentication is completed.

   Example
   ```
   sqlnet.cloud_user = dbuser1
   ```

4. Add a web server user password to the wallet.

   ```
   mkstore -wrl wallet_location  -createEntry username password
   ```
   Example
   ```
   mkstore -wrl  /app/wallet  -createEntry  dbuser1  Secretdb#
   ```

5. Make the wallet automatically log in and protect this wallet directory using operating system file permissions or any other means. Do this so that only the database client can have read access to it. Refer to the operating system utilities for information about changing the file permissions.

   ```
   orapki wallet create -wallet wallet_directory -auto_login
   ```
   Example
   ```
   orapki wallet create -wallet /app/wallet -auto_login
   ```

> **Note:**
>
> Oracle has introduced a new auto-login wallet version (7) with Oracle Database 23ai. Version 6 of the Oracle local auto-login wallet is deprecated.
>
> You can update your local auto-login wallet by modifying it with `orapki`.

6. Update the `sqlnet.ora` file with the wallet entry.

   Example

```
wallet_location=(SOURCE=(METHOD=file)(METHOD_DATA=(DIRECTORY=/app/
wallet)))
```

> **Note:**
>
> - The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.
>   For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.
> - The `mkstore` wallet management command line tool is deprecated with Oracle Database 23ai, and can be removed in a future release.
>
>   To manage wallets, Oracle recommends that you use the `orapki` command line tool.

Default

None

Related Topics

- *Oracle Database Security Guide*

5.2.42 SQLNET.COMPRESSION

Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION` to enable or disable data compression.

Purpose

To enable or disable data compression. If both the server and client have this parameter set to `ON`, then compression is used for the connection.

> **Note:**
>
> The `SQLNET.COMPRESSION` parameter applies to all database connections, except for Oracle Data Guard streaming redo and SecureFiles LOBs (Large Objects).

Default

off

Values

- `on` to enable data compression.
- `off` to disable data compression.

Example

```
SQLNET.COMPRESSION=on
```

5.2.43 SQLNET.COMPRESSION_ACCELERATION

Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_ACCELERATION` to specify the use of hardware accelerated version of compression using this parameter if it is available for that platform.

Purpose

To specify the use of hardware accelerated version of compression using this parameter if it is available for that platform.

Usage Notes

You can set this parameter in the Oracle Connection Manager alias description.

Default

on

Values

- `on`
- `off`
- `0`
- `1`

Example 5-4 Example

```
compression_acceleration = on
```

5.2.44 SQLNET.COMPRESSION_LEVELS

Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_LEVELS` to specify the compression level.

Purpose

To specify the compression level.

Usage Notes

The compression levels are used at the time of negotiation to verify which levels are used at both ends, and to select one level.

For Database Resident Connection Pooling (DRCP), only the compression level `low` is supported.

Default

low

Values

- `low` to use low CPU usage and low compression ratio
- `high` to use high CPU usage and high compression ratio

Example

```
SQLNET.COMPRESSION_LEVELS=(high)
```

5.2.45 SQLNET.COMPRESSION_THRESHOLD

Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_THRESHOLD` to specify the minimum data size for which compression is needed.

Purpose

To specify the minimum data size, in bytes, for which compression is needed.

Usage Notes

Compression is not to be performed if the size of the data you are sending is less than this value.

Default

1024 bytes

Example

```
SQLNET.COMPRESSION_THRESHOLD=1024
```

5.2.46 SQLNET.CRYPTO_CHECKSUM_CLIENT

Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_CLIENT` to specify the desired data integrity behavior when this client or server acting as a client connects to a server.

Purpose

To specify the checksum behavior for the client. The behavior partially depends on the `SQLNET.CRYPTO_CHECKSUM_SERVER` setting at the other end of the connection.

Default

`accepted`

Values

- `accepted` to enable the security service if required or requested by the other side
- `rejected` to disable the security service, even if required by the other side
- `requested` to enable the security service if the other side allows it
- `required` to enable the security service and disallow the connection if the other side is not enabled for the security service

Example

```
SQLNET.CRYPTO_CHECKSUM_CLIENT=accepted
```

5.2.47 SQLNET.CRYPTO_CHECKSUM_SERVER

Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_SERVER` to specify the data integrity behavior when a client or another server acting as a client connects to this server.

Purpose

To specify the checksum behavior for the database. The behavior partially depends on the `SQLNET.CRYPTO_CHECKSUM_CLIENT` setting at the other end of the connection.

Default

`accepted`

Values

- `accepted` to enable the security service if required or requested by the other side
- `rejected` to disable the security service, even if required by the other side
- `requested` to enable the security service if the other side allows it
- `required` to enable the security service and disallow the connection if the other side is not enabled for the security service

Example

```
SQLNET.CRYPTO_CHECKSUM_SERVER=accepted
```

5.2.48 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT

Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` to specify a list of data integrity algorithms that this client or server acting as a client uses.

Purpose

To specify a list of crypto-checksum algorithms for the client to use.

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed on this side is specified, the connection terminates with the `ORA-12650: No common encryption or data integrity algorithm error` error message.

Default

All available algorithms

Values

- `MD5` for the RSA Data Security MD5 algorithm

  The `MD5` algorithm is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

- `SHA1` for the Secure Hash Algorithm

  The use of SHA-1 with `DBMS_CRYPTO`, `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated.

  Using SHA-1 (Secure Hash Algorithm 1) with the parameters `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated in this release, and can be desupported in a future release. Using SHA-1 ciphers with `DBMS_CRYPTO` is also deprecated (`HASH_SH1`, `HMAC_SH1`). Instead of using SHA1, Oracle recommends that you start using a stronger SHA-2 cipher in place of the SHA-1 cipher.

- `SHA256` for SHA-2 uses 256 bits with the hashing algorithm
- `SHA384` for SHA-2 uses 384 bits with the hashing algorithm
- `SHA512` for SHA-2 uses 512 bits with the hashing algorithm

Example

```
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT=(SHA256, MD5)
```

5.2.49 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER

Use the `sqlnet.ora` parameter `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` to specify the data integrity algorithms that this server or client to another server uses, in order of intended use.

Purpose

To specify a list of crypto-checksum algorithms for the database to use.

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm is specified that is not installed on this side, the connection terminates with the `ORA-12650: No common encryption or data integrity algorithm error` error message.

Default

All available algorithms

Values

- `MD5` for the RSA Data Security's MD5 algorithm

  The `MD5` algorithm is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

- `SHA1` for the Secure Hash algorithm

  The use of SHA-1 with `DBMS_CRYPTO`, `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated.

  Using SHA-1 (Secure Hash Algorithm 1) with the parameters `SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT` and `SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER` is deprecated in this release, and can be desupported in a future release. Using SHA-1 ciphers with `DBMS_CRYPTO` is also deprecated (`HASH_SH1`, `HMAC_SH1`). Instead of using SHA1, Oracle recommends that you start using a stronger SHA-2 cipher in place of the SHA-1 cipher.

- `SHA256` for SHA-2 uses 256 bits with the hashing algorithm
- `SHA384` for SHA-2 uses 384 bits with the hashing algorithm
- `SHA512` for SHA-2 uses 512 bits with the hashing algorithm

Example

```
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER=(SHA256, MD5)
```

5.2.50 SQLNET.DBFW_PUBLIC_KEY

Use the `sqlnet.ora` parameter `SQLNET.DBFW_PUBLIC_KEY` to provide Oracle Database Firewall public keys to the Advanced Security Option (ASO) by specifying the file that stores the public keys.

Purpose

To provide Oracle Database Firewall public keys to Advanced Security Option (ASO) by specifying the name of the file that stores the Oracle Database Firewall public keys.

Default

None

Values

Full path name of the operating system file that has the public keys

Example

```
SQLNET.DBFW_PUBLIC_KEY="/path_to_file/dbfw_public_key_file.txt"
```

> **See Also:**
>
> "SQLNET.ENCRYPTION_TYPES_SERVER"

5.2.51 SQLNET.DOWN_HOSTS_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.DOWN_HOSTS_TIMEOUT` to specify the amount of time in seconds that server hosts down state information remains in the client cache.

Purpose

To specify the amount of time in seconds that information about the `down` state of server hosts is kept in the client process cache.

Usage Notes

Clients discover the `down` state of server hosts when attempting connections. When a connection attempt fails, the information about the `down` state of the server host is added to the client process cache. Subsequent connection attempts by the same client process move the addresses of the `down` hosts to the end of the address list, thereby reducing the priority of `down` hosts. When the duration of time that is specified by the `SQLNET.DOWN_HOSTS_TIMEOUT` parameter has elapsed, the host is purged from the process cache and its priority in the address list is restored.

Default

600 seconds (10 minutes)

Values

Any positive integer

Example

```
SQLNET.DOWN_HOSTS_TIMEOUT=60
```

5.2.52 SQLNET.ENCRYPTION_CLIENT

Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_CLIENT` to set the encryption behavior when this client or server acting as a client connects to a server.

Purpose

To enable encryption for clients. Setting the `tnsnames.ora` parameter `IGNORE_ANO_ENCRYPTION_FOR_TCPS` to `TRUE` disables `SQLNET.ENCRYPTION_CLIENT`.

The behavior of the client partially depends on the value set for `SQLNET.ENCRYPTION_SERVER` at the other end of the connection.

Default

accepted

Values

- accepted to enable the security service if required or requested by the other side
- rejected to disable the security service, even if required by the other side
- requested to enable the security service if the other side allows it
- required to enable the security service and disallow the connection if the other side is not enabled for the security service

Example

```
SQLNET.ENCRYPTION_CLIENT=accepted
```

5.2.53 SQLNET.ENCRYPTION_SERVER

The sqlnet.ora parameter SQLNET.ENCRYPTION_SERVER specifies the encryption behavior when a client or a server acting as a client connects to this server.

Purpose

To enable encryption for the database. Setting SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS to FALSE disables SQLNET.ENCRYPTION_SERVER.

The behavior of the server partially depends on the SQLNET.ENCRYPTION_CLIENT setting at the other end of the connection.

Default

accepted

Values

- accepted to enable the security service if required or requested by the other side
- rejected to disable the security service, even if required by the other side
- requested to enable the security service if the other side allows it
- required to enable the security service and disallow the connection if the other side is not enabled for the security service

Example

```
SQLNET.ENCRYPTION_SERVER=accepted
```

5.2.54 SQLNET.ENCRYPTION_TYPES_CLIENT

Use the `sqlnet.ora` parameter `SQLNET.ENCRYPTION_TYPES_CLIENT` to specify the encryption algorithms this client or the server acting as a client uses.

Purpose

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed is specified on this side, the connection terminates with the `ORA-12650: No common encryption or data integrity algorithm` error message.

Usage Notes

Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated.

The deprecated algorithms for `DBMS_CRYPTO` and native network encryption include MD4, MD5, DES, 3DES, and RC4-related algorithms as well as 3DES for Transparent Data Encryption (TDE). Removing older, less secure cryptography algorithms prevents accidental use of these algorithms. To meet your security requirements, Oracle recommends that you use more modern cryptography algorithms, such as the Advanced Encryption Standard (AES).

As a consequence of this deprecation, Oracle recommends that you review your network encryption configuration to see if you have specified use of any of the deprecated algorithms. If any are found, then switch to using a more modern cipher, such as AES. Also, if you are currently using 3DES encryption for your TDE deployment, then you should plan to migrate to a more modern algorithm such as AES. For more information, refer to *Oracle Database Security Guide*

To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

Default

All available algorithms

Values

Approved algorithms:

- `AES128` for AES (128-bit key size)
- `AES192` for AES (192-bit key size)
- `AES256` for AES (256-bit key size)

Deprecated algorithms:

- `3DES112` for triple DES with a two-key (112-bit) option
- `3DES168` for triple DES with a three-key (168-bit) option
- `DES` for standard DES (56-bit key size)
- `DES40` for DES (40-bit key size)
- `RC4_40` for RSA RC4 (40-bit key size)

- RC4_56 for RSA RC4 (56-bit key size)
- RC4_128 for RSA RC4 (128-bit key size)
- RC4_256 for RSA RC4 (256-bit key size)

Example

```
SQLNET.ENCRYPTION_TYPES_CLIENT=(AES256)
```

5.2.55 SQLNET.ENCRYPTION_TYPES_SERVER

Use the sqlnet.ora parameter SQLNET.ENCRYPTION_TYPES_SERVER to specify the encryption algorithms this server uses in the order of the intended use.

Purpose

This list is used to negotiate a mutually acceptable algorithm with the client end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm that is not installed is specified on this side, the connection terminates with an ORA-12650: No common encryption or data integrity algorithm error message.

Default

All available algorithms

Values

Approved algorithms:

- AES128 for AES (128-bit key size)
- AES192 for AES (192-bit key size)
- AES256 for AES (256-bit key size)

Deprecated algorithms:

- 3DES112 for triple DES with a two-key (112-bit) option
- 3DES168 for triple DES with a three-key (168-bit) option
- DES for standard DES (56-bit key size)
- DES40 for DES40 (40-bit key size)
- RC4_40 for RSA RC4 (40-bit key size)
- RC4_56 for RSA RC4 (56-bit key size)
- RC4_128 for RSA RC4 (128-bit key size)
- RC4_256 for RSA RC4 (256-bit key size)

Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated.

The deprecated algorithms for DBMS_CRYPTO and native network encryption include MD4, MD5, DES, 3DES, and RC4-related algorithms as well as 3DES for Transparent Data Encryption (TDE). Removing older, less secure cryptography algorithms prevents accidental use of these algorithms. To meet your security requirements, Oracle recommends that you use more modern cryptography algorithms, such as the Advanced Encryption Standard (AES).

As a consequence of this deprecation, Oracle recommends that you review your network encryption configuration to see if you have specified use of any of the deprecated algorithms. If any are found, then switch to using a more modern cipher, such as AES. Also, if you are currently using 3DES encryption for your TDE deployment, then you should plan to migrate to a more modern algorithm such as AES. For more information, refer to *Oracle Database Security Guide*

To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

Example

```
SQLNET.ENCRYPTION_TYPES_SERVER=(AES256, AES192, ...)
```

5.2.56 SQLNET.EXPIRE_TIME

Use the `sqlnet.ora` parameter `SQLNET.EXPIRE_TIME` to specify how often, in minutes, to verify that client and server connections are active.

Purpose

To specify time intervals, in minutes, for how often to verify that client and server connections are active.

Usage Notes

Setting a value greater than 0 ensures that connections are not left open indefinitely due to an unusual client termination. If your environment supports TCP keepalive tuning, then Oracle Net Services automatically uses the enhanced detection model and tunes the TCP keepalive parameters.

If the verification check identifies a terminated connection or a connection that is no longer in use, then the check returns an error, causing the server process to exit.

The `sqlnet.ora` parameter `SQLNET.EXPIRE_TIME` is primarily intended for the database server, which typically handles multiple connections simultaneously.

You can also use this parameter for database clients to verify if the server connection is active.

Limitations on using the terminated connection detection feature are:

- You cannot use it on bequeathed connections.
- Though very small, a probe packet generates additional traffic that may degrade your network performance.
- Depending on your operating system, the server may need to perform additional processing to distinguish the connection probing event from other events. This can also result in degraded network performance.

Default

0

Minimum Value

0

Recommended Value

10

Example

```
SQLNET.EXPIRE_TIME=10
```

5.2.57 SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS

Use the `sqlnet.ora` parameter `SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS` to ignore the value that is set for the parameter `SQLNET.ENCRYPTION_SERVER` for TCPS connections. This disables ANO encryption on the TCPS listener.

Purpose

Use `SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS` on your server to ignore the value that is set for `SQLNET.ENCRYPTION_SERVER` for TCPS connections. Doing this disables ANO encryption on the TCPS listener.

Default

```
FALSE
```

Example 5-5 Example

```
SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS=TRUE
```

5.2.58 SQLNET.INBOUND_CONNECT_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.INBOUND_CONNECT_TIMEOUT` to specify the amount of time that clients have to connect with the database and authenticate.

Purpose

Use the parameter `SQLNET.INBOUND_CONNECT_TIMEOUT` to specify the time limit in `ms`, `sec`, or `min`, within which a client must connect with the database and provide authentication information.

Usage Notes

If the client fails to connect and complete the authentication within the specified timeframe, then the database terminates the connection. In addition, the database logs the IP address of the client and writes an ORA-12170 error message to the database alert log file.

The client receives either an `ORA-12547: TNS:lost contact` or an `ORA-12637: Packet receive failed` error message.

The default value of `SQLNET.INBOUND_CONNECT_TIMEOUT` is appropriate for typical scenarios. However, if you need to set a different value, then Oracle recommends setting this parameter in combination with the`INBOUND_CONNECT_TIMEOUT_`*`listener_name`* parameter in the `listener.ora` file. When specifying the values for these parameters, note the following recommendations:

- Set both parameters to a low value initially.

- Set the value of the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter to a lower value than the value that you have set for the SQLNET.INBOUND_CONNECT_TIMEOUT parameter.

It accepts different timeouts with or without space between the value and the unit. If you do not set a unit of measurement for SQLNET.INBOUND_CONNECT_TIMEOUT, then the default unit is sec. For example, you can set INBOUND_CONNECT_TIMEOUT_*listener_name* to 2 seconds and set the SQLNET.INBOUND_CONNECT_TIMEOUT parameter to 3 seconds. If clients are unable to complete the connections within the specified time due to system or network delays that are normal for the particular environment, then increase the value for SQLNET.INBOUND_CONNECT_TIMEOUT as needed.

Default

60 seconds

Example

```
SQLNET.INBOUND_CONNECT_TIMEOUT=3ms
```

5.2.59 SQLNET.FALLBACK_AUTHENTICATION

Use the sqlnet.ora parameter SQLNET.FALLBACK_AUTHENTICATION to specify whether to attempt password-based authentication if Kerberos authentication fails.

Purpose

To specify whether to attempt to use password-based authentication if Kerberos authentication fails. This is relevant for direct connections as well as database link connections.

Default

```
FALSE
```

Example

```
SQLNET.FALLBACK_AUTHENTICATION=TRUE
```

> **See Also:**
>
> *Oracle Database Security Guide*

5.2.60 SQLNET.KERBEROS5_CC_NAME

Use the sqlnet.ora parameter SQLNET.KERBEROS5_CC_NAME to specify the complete path name to the Kerberos credentials cache (CC) file.

Purpose

To specify the complete path name to the Kerberos CC file.

Usage Notes

- In addition to the `sqlnet.ora` file, you can set this parameter in the connect string or `tnsnames.ora` file. Note that this parameter is called `KERBEROS5_CC_NAME` in the `tnsnames.ora` or connect string. The connect string value takes precedence.
- This parameter supports multiple principals for the storage of credentials that are returned by KDC in an encrypted format.

  You can use the `okinit`, `oklist`, and `okdstry` utilities to configure encrypted cache files for all Kerberos principals. These utilities work with encrypted cache files only if you specify the cache path using `SQLNET.KERBEROS5_CC_NAME`.
- `SQLNET.KERBEROS5_CC_NAME` is mandatory for all additional Kerberos users and principals. Optionally, you can set the `KERBEROS5_PRINCIPAL` parameter to specify the Kerberos principal name associated with the credential cache (specified through `SQLNET.KERBEROS5_CC_NAME`). You can set `KERBEROS5_PRINCIPAL` in the connect string, `sqlnet.ora` file, or `tnsnames.ora` file.

  Oracle Database checks `KERBEROS5_PRINCIPAL` against the value that is retrieved from the credential cache. If the two values do not match, then the user is not authenticated.

Values and Examples

You can use the following formats to specify a value for `SQLNET.KERBEROS5_CC_NAME`:

- If the Oracle database is using a directory cache:
  - `SQLNET.KERBEROS5_CC_NAME=`*`complete_path_to_cc_file`*

    For example:

    ```
    SQLNET.KERBEROS5_CC_NAME=/tmp/kcache
    ```

    ```
    SQLNET.KERBEROS5_CC_NAME=D:\tmp\kcache
    ```
  - `SQLNET.KERBEROS5_CC_NAME=FILE:`*`complete_path_to_cc_ file`*

    For example:

    ```
    SQLNET.KERBEROS5_CC_NAME=FILE:/tmp/kcache
    ```
- If the Oracle database is using the native Windows cache:
  - `SQLNET.KERBEROS5_CC_NAME=OSMSFT://`
  - `SQLNET.KERBEROS5_CC_NAME=MSLSA:`

  The `OSMSFT` and `MSLSA` options specify that the file is on Microsoft Windows and is running Microsoft Kerberos Key Distribution Center (KDC).

Default

The default value is operating system-dependent, as follows:

- On Linux and UNIX operating systems: `/tmp/krb5cc_userid`
- On Microsoft Windows operating systems: `c:\tmp\krbcache`

Related Topics

- KERBEROS5_PRINCIPAL
  Use the `KERBEROS5_PRINCIPAL` parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.

- KERBEROS5_CC_NAME
  Use the `tnsnames.ora` parameter `KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- *Oracle Database Security Guide*

5.2.61 SQLNET.KERBEROS5_CLOCKSKEW

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CLOCKSKEW` to specify how much time elapses before a Kerberos credential is considered out-of-date.

Purpose

To specify how many seconds elapse before a Kerberos credential is considered out-of-date.

Default

300

Example

```
SQLNET.KERBEROS5_CLOCKSKEW=1200
```

> **See Also:**
>
> *Oracle Database Security Guide*

5.2.62 SQLNET.KERBEROS5_CONF

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CONF` to specify the path name to the Kerberos configuration file that contains the realm for the default Key Distribution Center (KDC) and that maps realms to KDC hosts.

Purpose

To specify the complete path name to the Kerberos configuration file that contains the realm for the default Key Distribution Center (KDC) and that also maps realms to KDC hosts.

Usage Notes

KDC maintains a list of user principals and is contacted through the `kinit` program for the user's initial ticket.

The `AUTO_DISCOVER` option enables the automatic discovery of KDC and its realms. It is the default configuration for Kerberos clients. If there are multiple realms to specify, then Oracle recommends creating configuration files instead of using the `AUTO_DISCOVER` option. This option is supported for all operating systems with such a feature.

Default

`/krb5/krb.conf` on Linux and UNIX operating systems

`c:\krb5\krb.conf` on Microsoft Windows operating systems

Values

- Directory path to `krb.conf` file
- `AUTO_DISCOVER`

Example

```
SQLNET.KERBEROS5_CONF=/krb5/krb.conf
```

> **See Also:**
>
> *Oracle Database Security Guide*

5.2.63 SQLNET.KERBEROS5_CONF_LOCATION

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CONF_LOCATION` to specify the directory for the Kerberos configuration file. The `SQLNET.KERBEROS5_CONF_LOCATION` parameter also specifies that the file is created by the system and not by the client.

Purpose

To specify the directory for the Kerberos configuration file. The parameter also specifies that the file is created by the system, and not by the client.

Usage Notes

The configuration file uses DNS look-up to obtain the realm for the default KDC, and it maps realms to the KDC hosts. This option is supported for all operating systems that support this feature.

Default

`/krb5` on Linux and UNIX operating systems

`c:\krb5` on Microsoft Windows operating systems

Example

SQLNET.KERBEROS5_CONF_LOCATION=/krb5

5.2.64 SQLNET.KERBEROS5_KEYTAB

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_KEYTAB` to specify the path name to the Kerberos principal or, secret, key mapping file that extracts keys and decrypts incoming authentication information.

Purpose

To specify the complete path name to the Kerberos principal or, secret, key mapping file that extracts keys and decrypts incoming authentication information.

Default

`/etc/v5srvtab` on Linux and UNIX operating systems

`c:\krb5\v5srvtab` on Microsoft Windows operating systems

Example

```
SQLNET.KERBEROS5_KEYTAB=/etc/v5srvtab
```

> **See Also:**
>
> *Oracle Database Security Guide*

5.2.65 SQLNET.KERBEROS5_REALMS

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_REALMS` to specify the complete path name to the Kerberos realm translation file that maps a host name or domain name to a realm.

Purpose

To specify the complete path name to the Kerberos realm translation file that maps a host name or domain name to a realm.

Default

`/krb5/krb.realms` on Linux and UNIX operating systems

`c:\krb5\krb.realms` on Microsoft Windows operating systems

Example

```
SQLNET.KERBEROS5_REALMS=/krb5/krb.realms
```

> **See Also:**
>
> *Oracle Database Security Guide*

5.2.66 SQLNET.KERBEROS5_REPLAY_CACHE

Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_REPLAY_CACHE` to specify that the replay cache is stored in operating system-managed memory on the server, and that file-based replay cache is not used.

Purpose

To specify that replay cache is stored in operating system-managed memory on the server and that file-based replay cache is not used.

Usage Notes

The `OS_MEMORY` option specifies that the replay cache is stored in operating system-managed memory on the server, and file-based replay cache is not used.

Example

```
SQLNET_KERBEROS5_REPLAY_CACHE=OS_MEMORY
```

5.2.67 SQLNET.OUTBOUND_CONNECT_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.OUTBOUND_CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.

Purpose

To specify the time in ms, sec, or min for clients to establish an Oracle Net connection to the database instance.

Usage Notes

If an Oracle Net connection is not established in the time specified, then the connection attempt is terminated. The client receives the following error:

`ORA-12170: Cannot connect. Outbound connect timeout of` *`time_interval`* `for` *`host_port or key`*`. (CONNECTION_ID=`*`ID_string`*`).`

The outbound connect timeout interval is a superset of the TCP connect timeout interval that specifies a limit on the time needed to establish a TCP connection. Additionally, the outbound connect timeout interval includes the time taken to be connected to an Oracle instance that is providing the service. It accepts different timeouts with or without space between the value and the unit.

Without this parameter, a client connection request to the database server may be blocked for the default TCP connect timeout duration (60 `seconds`) when the database server host system is unreachable. In this case, no unit is mentioned and the default unit is `sec`.

The outbound connect timeout interval is only applicable for TCP, TCP with TLS, and IPC transport connections.

This parameter is overridden by the `CONNECT_TIMEOUT` parameter in the address description.

Default

None

Example

```
SQLNET.OUTBOUND_CONNECT_TIMEOUT=10 ms
```

Related Topics

- CONNECT_TIMEOUT
  Use the `tnsnames.ora` parameter `CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.

5.2.68 SQLNET.RADIUS_ALLOW_WEAK_CLIENTS

Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.

Purpose

To control the transport protocol that the Oracle Database client must use for communication between the database client and database server, if the database client wants to use RADIUS authentication.

The default value is `FALSE` so that database clients can connect to the database server (to use RADIUS authentication) only if the connecting protocol used is TCPS.

Usage Notes

- Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

  Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

  The updated RADIUS API uses TCPS as the protocol for secure communication.

- Starting with Oracle Database 23ai, users authenticating to the database using the legacy RADIUS API no longer are granted administrative privileges.

  In previous releases, users authenticating with RADIUS API could be granted administrative privileges such as `SYSDBA` or `SYSBACKUP`. In Oracle Database 23ai, Oracle introduces a new RADIUS API that uses the latest standards. To grant administrative privileges to users, ensure the database connection to the database uses the new RADIUS API, and that you are using the Oracle Database 23ai client to connect to the Oracle Database 23ai server.

Values

- `TRUE`: To allow database clients to connect using a weak protocol, such as User Datagram Protocol (UDP).

- FALSE: To allow database clients to connect using only a strong protocol, such as TCPS.

Default

FALSE

Example

```
SQLNET.RADIUS_ALLOW_WEAK_CLIENTS=FALSE
```

Related Topics

- *Oracle Database Security Guide*

5.2.69 SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL

Use the server-side sqlnet.ora parameter SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL to allow weak Oracle Database clients to use RADIUS authentication.

Purpose

To allow weak Oracle Database clients, which use non-TCPS protocol for connecting to the Oracle Database server, to use RADIUS authentication.

The default value is FALSE so that only strong clients (using TCPS for connecting to the database server) can use RADIUS authentication.

Usage Notes

- Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

  Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: SQLNET.RADIUS_ALTERNATE, SQLNET.RADIUS_ALTERNATE_PORT, SQLNET.RADIUS_AUTHENTICATION, and SQLNET.RADIUS_AUTHENTICATION_PORT. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

  The updated RADIUS API uses TCPS as the protocol for secure communication.

- Starting with Oracle Database 23ai, users authenticating to the database using the legacy RADIUS API no longer are granted administrative privileges.

  In previous releases, users authenticating with RADIUS API could be granted administrative privileges such as SYSDBA or SYSBACKUP. In Oracle Database 23ai, Oracle introduces a new RADIUS API that uses the latest standards. To grant administrative privileges to users, ensure the database connection to the database uses the new RADIUS API, and that you are using the Oracle Database 23ai client to connect to the Oracle Database 23ai server.

Values

- TRUE: To allow weak database clients to use RADIUS authentication
- FALSE: To allow only strong database clients (and block weak clients) to use RADIUS authentication

Default

FALSE

Example

```
SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL=FALSE
```

Related Topics

- *Oracle Database Security Guide*

5.2.70 SQLNET.RADIUS_ALTERNATE

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE` to specify an alternate RADIUS server to be used when the primary server is unavailable.

Purpose

To specify the location of an alternate RADIUS server to be used for fault tolerance when the primary server is unavailable. The value can be either the IP address or host name of the server.

Usage Notes

Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

If your database server supports the updated RADIUS standards, then use the `SQLNET.RADIUS_ALTERNATE_TLS_HOST` parameter instead of the deprecated `SQLNET.RADIUS_ALTERNATE` parameter.

If you need to enable pre-release 23ai clients to connect RADIUS users using the older RADIUS standards (which are blocked by default), then you must set one or both of the `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` and `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` parameters.

Syntax

`SQLNET.RADIUS_ALTERNATE=`*`(hostname_or_IP_address_of_alternate_RADIUS_server)`*

Default

None

Example

```
SQLNET.RADIUS_ALTERNATE=(radius-server2)
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.RADIUS_ALTERNATE_TLS_HOST
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_HOST` to specify the host name of an alternate RADIUS server to be used when the primary server is unavailable.
- SQLNET.RADIUS_ALLOW_WEAK_CLIENTS
  Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.
- SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` to allow weak Oracle Database clients to use RADIUS authentication.

5.2.71 SQLNET.RADIUS_ALTERNATE_PORT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_PORT` to specify the listening port of an alternate RADIUS server.

Purpose

To specify the listening port of an alternate RADIUS server.

Usage Notes

Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

If your database server supports the updated RADIUS standards, then use the `SQLNET.RADIUS_ALTERNATE_TLS_PORT` parameter instead of the deprecated `SQLNET.RADIUS_ALTERNATE_PORT` parameter.

If you need to enable pre-release 23ai clients to connect RADIUS users using the older RADIUS standards (which are blocked by default), then you must set one or both of the `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` and `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` parameters.

Syntax

```
SQLNET.RADIUS_ALTERNATE_PORT=(listening_port_of_alternate_RADIUS_server)
```

Default

```
1812
```

Example

```
SQLNET.RADIUS_ALTERNATE_PORT=(1667)
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.RADIUS_ALTERNATE_TLS_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_PORT` to specify the listening port of an alternate RADIUS server.
- SQLNET.RADIUS_ALLOW_WEAK_CLIENTS
  Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.
- SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` to allow weak Oracle Database clients to use RADIUS authentication.

5.2.72 SQLNET.RADIUS_ALTERNATE_RETRIES

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_RETRIES` to specify the number of times that the database resends messages to alternate RADIUS servers.

Purpose

To specify the number of times that the database server should resend messages to an alternate RADIUS server.

Default

```
3
```

Example

```
SQLNET.RADIUS_ALTERNATE_RETRIES=4
```

5.2.73 SQLNET.RADIUS_ALTERNATE_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TIMEOUT` to set the time for an alternate RADIUS server to wait for a response.

Purpose

To set the time, in seconds, for an alternate RADIUS server to wait for a response.

Syntax

```
SQLNET.RADIUS_ALTERNATE_TIMEOUT=time_in_seconds
```

Default

```
5
```

Example

```
SQLNET.RADIUS_ALTERNATE_TIMEOUT=5
```

Related Topics

- *Oracle Database Security Guide*

5.2.74 SQLNET.RADIUS_ALTERNATE_TLS_HOST

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_HOST` to specify the host name of an alternate RADIUS server to be used when the primary server is unavailable.

Purpose

To specify the host name of an alternate RADIUS server, which is used for fault tolerance when the primary server is unavailable.

Usage Notes

Use this parameter only if your RADIUS server implements RADIUS with TLS over TCP.

Syntax

```
SQLNET.RADIUS_ALTERNATE_TLS_HOST=(TLS_hostname_of_alternate_RADIUS_server)
```

Default

None

Example

```
SQLNET.RADIUS_ALTERNATE_TLS_HOST=(radius-server2)
```

Related Topics

- *Oracle Database Security Guide*

5.2.75 SQLNET.RADIUS_ALTERNATE_TLS_PORT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_ALTERNATE_TLS_PORT` to specify the listening port of an alternate RADIUS server.

Purpose

To specify the listening port of an alternate RADIUS server. The default port is `2083`. If the alternate server uses a different port, then specify that value.

Usage Notes

Use this parameter only if your RADIUS server implements RADIUS with TLS over TCP.

Syntax

SQLNET.RADIUS_ALTERNATE_TLS_PORT=(*listening_TLS_port_of_alternate_RADIUS_server*)

Default

2083

Example

```
SQLNET.RADIUS_ALTERNATE_TLS_PORT=(5530)
```

Related Topics

- *Oracle Database Security Guide*

5.2.76 SQLNET.RADIUS_AUTHENTICATION

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION` to specify the location of a primary RADIUS server.

Purpose

To specify the location of a primary RADIUS server. The value can be either the IP address or host name of the server.

Usage Notes

Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

If your database server supports the updated RADIUS standards, then use the `SQLNET.RADIUS_AUTHENTICATION_TLS_HOST` parameter instead of the deprecated `SQLNET.RADIUS_AUTHENTICATION` parameter.

If you need to enable pre-release 23ai clients to connect RADIUS users using the older RADIUS standards (which are blocked by default), then you must set one or both of the `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` and `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` parameters.

Syntax

SQLNET.RADIUS_AUTHENTICATION=(*hostname_or_IP_address_of_primary_RADIUS_server*)

Default

Local host

Example

```
SQLNET.RADIUS_AUTHENETICATION=(radius-server1)
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.RADIUS_AUTHENTICATION_TLS_HOST
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_HOST` to specify the host name of a primary RADIUS server.
- SQLNET.RADIUS_ALLOW_WEAK_CLIENTS
  Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.
- SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` to allow weak Oracle Database clients to use RADIUS authentication.

5.2.77 SQLNET.RADIUS_AUTHENTICATION_INTERFACE

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_INTERFACE` to specify the class that contains the user interface for interacting with users.

Purpose

To specify the class containing the user interface that is used to interact with the user.

Default

```
DefaultRadiusInterface
```

Example

```
SQLNET.RADIUS_AUTHENTICATION_INTERFACE=DefaultRadiusInterface
```

5.2.78 SQLNET.RADIUS_AUTHENTICATION_PORT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_PORT` to specify the listening port of a primary RADIUS server.

Purpose

To specify the listening port of a primary RADIUS server.

Usage Notes

Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

If your database server supports the updated RADIUS standards, then use the `SQLNET.RADIUS_AUTHENTICATION_TLS_PORT` parameter instead of the deprecated `SQLNET.RADIUS_AUTHENTICATION_PORT` parameter.

If you need to enable pre-release 23ai clients to connect RADIUS users using the older RADIUS standards (which are blocked by default), then you must set one or both of the `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` and `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` parameters.

Syntax

`SQLNET.RADIUS_AUTHENTICATION_PORT=(`*`listening_port_of_primary_RADIUS_server`*`)`

Default

`1645`

Example

```
SQLNET.RADIUS_AUTHENTICATION_PORT=(1667)
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.RADIUS_AUTHENTICATION_TLS_PORT
  Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_PORT` to specify the listening port of a primary RADIUS server.
- SQLNET.RADIUS_ALLOW_WEAK_CLIENTS
  Use the client-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_CLIENTS` to control the transport protocol that the Oracle Database client must use for communicating with the Oracle Database server.
- SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL
  Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_ALLOW_WEAK_PROTOCOL` to allow weak Oracle Database clients to use RADIUS authentication.

5.2.79 SQLNET.RADIUS_AUTHENTICATION_RETRIES

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_RETRIES` to specify the number of times the database should resend messages to a primary RADIUS server.

Purpose

To specify the number of times the database should resend messages to a primary RADIUS server.

Default

`3`

Example

```
SQLNET.RADIUS_AUTHENTICATION_RETRIES=4
```

5.2.80 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TIMEOUT` to specify the amount of time that the database should wait for a response from a primary RADIUS server.

Purpose

To specify the amount of time, in seconds, that the database should wait for a response from a primary RADIUS server.

Default

`5`

Example

```
SQLNET.RADIUS_AUTHENTICATION_TIMEOUT=10
```

5.2.81 SQLNET.RADIUS_AUTHENTICATION_TLS_HOST

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_HOST` to specify the host name of a primary RADIUS server.

Purpose

To specify the host name of a primary RADIUS server. This value is mandatory. If you do not set this parameter, then authentication fails.

Usage Notes

Use this parameter only if your RADIUS server implements RADIUS with TLS over TCP.

Syntax

```
SQLNET.RADIUS_AUTHENTICATION_TLS_HOST=(TLS_hostname_of_primary_RADIUS_server)
```

Default

None

Example

```
SQLNET.RADIUS_AUTHENTICATION_TLS_HOST=(radius-server1)
```

Related Topics

- *Oracle Database Security Guide*

5.2.82 SQLNET.RADIUS_AUTHENTICATION_TLS_PORT

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_AUTHENTICATION_TLS_PORT` to specify the listening port of a primary RADIUS server.

Purpose

To specify the listening port of a primary RADIUS server. The default port is `2083`. If the server uses a different port, then specify that value.

Usage Notes

Use this parameter only if your RADIUS server implements RADIUS with TLS over TCP.

Syntax

```
SQLNET.RADIUS_AUTHENTICATION_TLS_PORT=(listening_TLS_port_of_primary_RADIUS_server)
```

Default

```
2083
```

Example

```
SQLNET.RADIUS_AUTHENTICATION_TLS_PORT=(5530)
```

Related Topics

- *Oracle Database Security Guide*

5.2.83 SQLNET.RADIUS_CHALLENGE_KEYWORD

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CHALLENGE_KEYWORD` to set the keyword for requesting a challenge from the RADIUS server.

Purpose

To set the keyword for requesting a challenge from the RADIUS server. By setting the challenge keyword, you let the user avoid using a password on the client to verify identity.

Syntax

```
SQLNET.RADIUS_CHALLENGE_KEYWORD=keyword
```

Default

```
challenge
```

Example

```
SQLNET.RADIUS_CHALLENGE_KEYWORD=challenge
```

Related Topics

- *Oracle Database Security Guide*

5.2.84 SQLNET.RADIUS_CHALLENGE_RESPONSE

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CHALLENGE_RESPONSE` to enable or disable challenge responses.

Purpose

To turn the challenge responses on or off.

Default

```
off
```

Values

```
on | off
```

Example

```
SQLNET.RADIUS_CHALLENGE_RESPONSE=on
```

5.2.85 SQLNET.RADIUS_CLASSPATH

Use the `sqlnet.ora` parameter `SQLNET.RADIUS_CLASSPATH` to set the path for Java classes and JDK Java libraries.

Purpose

To set the path for Java classes for a graphical interface, and to set the path to JDK Java libraries.

If you use the challenge-response authentication mode, then RADIUS displays a Java-based graphical interface. This interface first requests a password and then additional information, for example, a dynamic password that the user obtains from a token card.

Syntax

```
SQLNET.RADIUS_CLASSPATH=path_to_GUI_Java_classes
```

Default

$ORACLE_HOME/jlib/netradius.jar:$ORACLE_HOME/JRE/lib/sparc/native_threads

Example

```
SQLNET.RADIUS_CLASSPATH=/jre1.1
```

Related Topics

- *Oracle Database Security Guide*

5.2.86 SQLNET.RADIUS_SECRET

Use the sqlnet.ora parameter SQLNET.RADIUS_SECRET to specify the location of a RADIUS secret key.

Purpose

To specify the location of a RADIUS secret key.

Usage Notes

For RADIUS with TLS over TCP, the default value is radsec. This value is used if you do not set this parameter in the sqlnet.ora file.

There is no default value for RADIUS with UDP. You must configure this parameter with a directory path to the file containing secret key. For example:

ORACLE_HOME/network/security/radius.key

Example

```
SQLNET.RADIUS_SECRET=oracle/bin/admin/radiuskey
```

Related Topics

- *Oracle Database Security Guide*

5.2.87 SQLNET.RADIUS_SEND_ACCOUNTING

Use the sqlnet.ora parameter SQLNET.RADIUS_SEND_ACCOUNTING to enable and disable accounting.

Purpose

To turn accounting ON and OFF. When you enable accounting, packets are sent to the active RADIUS server at the listening port number's value plus one.

Default

OFF

Values

`ON | OFF`

Example

```
SQLNET.RADIUS_SEND_ACCOUNTING=ON
```

Related Topics

- *Oracle Database Security Guide*

5.2.88 SQLNET.RADIUS_TRANSPORT_PROTOCOL

Use the server-side `sqlnet.ora` parameter `SQLNET.RADIUS_TRANSPORT_PROTOCOL` to control the transport protocol that the Oracle Database server must use for communicating with the RADIUS server.

Purpose

To specify mutual Transport Layer Security (mTLS), Transport Layer Security (TLS), or User Datagram Protocol (UDP) as the protocol for communication between the Oracle Database server (acting as the RADIUS client) and the RADIUS server.

Usage Notes

- Starting with Oracle Database 23ai, the older RADIUS API that is based on Request for Comments (RFC) 2138 is deprecated.

  Oracle Database 23ai introduces an updated RADIUS API based on RFC 6613 and RFC 6614. Oracle recommends that you start planning on migrating to use the new RADIUS API as soon as possible. The new API is enabled by default. These parameters associated with the older RADIUS API are also deprecated: `SQLNET.RADIUS_ALTERNATE`, `SQLNET.RADIUS_ALTERNATE_PORT`, `SQLNET.RADIUS_AUTHENTICATION`, and `SQLNET.RADIUS_AUTHENTICATION_PORT`. Refer to the Radius API documentation for information on changing the default to use the older RADIUS API.

- Both the mTLS and TLS protocols implement the latest RADIUS API standards and enforce stronger security.

  When set to `MTLS`, a mutual or two-way TLS connection is established between the Oracle Database server and RADIUS server. You must configure an Oracle wallet on the database server to use mTLS. Ensure that the wallet stores RADIUS client user certificates and trusted CA certificates of both the RADIUS client and RADIUS server.

  When set to `TLS`, a one-way TLS connection is established between the Oracle Database server and RADIUS server. For walletless TLS connections (which do not use a client wallet), the RADIUS client automatically picks up common root certificates from the system default certificate store to verify the RADIUS server certificates. Use this value if your RADIUS server supports TLS (RADIUS over TCP) or TCPS (RADIUS with TLS over TCP).

- If you must use RADIUS with UDP for backward compatibility, then set this parameter to `UDP`. However, note that RADIUS with UDP uses the older RADIUS API standards and is considered insecure.

- If you omit this parameter value, then the default protocol, mTLS, is used.

Values

`MTLS | TLS | UDP`

Default

`MTLS`

Example

```
SQLNET.RADIUS_TRANSPORT_PROTOCOL=MTLS
```

Related Topics

- *Oracle Database Security Guide*
- *Oracle Database Net Services Administrator's Guide*

5.2.89 SQLNET.RECV_TIMEOUT

Use the `sqlnet.ora` parameter `SQLNET.RECV_TIMEOUT` to specify the duration of time that a database client or server should wait for data from a peer after establishing a connection.

Purpose

To specify the time for a database client or server to wait for data from the peer after establishing a connection. The peer must send data within the time interval that you specify.

You can specify the time in hours, minutes, seconds, or milliseconds by using the `hr`, `min`, `sec`, or `ms` keyword respectively. If you do not specify a unit of measurement, then the default unit is `sec`.

Usage Notes

Setting this parameter for clients ensures that receive operations are not left in a wait state indefinitely or for a long period due to an unusual termination of the server process or server busy state. If a client does not receive response data in the time specified, then the client logs `ORA-12535: TNS:operation timed out` and `ORA-12609: TNS: Receive timeout occurred` messages to the `sqlnet.log` file. If you set the value, then set the value initially to a low value and adjust the value according to the system and network capacity. If necessary, use this parameter with the `SQLNET.SEND_TIMEOUT` parameter.

You can also set this parameter on the server-side to specify the time, in `ms`, `sec`, or `min`, for a server to wait for client data after a connection is established. If a client does not send data in time specified, then the database server logs `ORA-12535: TNS:operation timed out` and `ORA-12609: TNS: Receive timeout occurred` messages to the `sqlnet.log` file. Without this parameter, the database server might continue to wait for data from clients that may be down or are experiencing problems. The server usually blocks input from the client and gets these timeouts frequently if you set it to a low value.

Default

None

Minimum Value

1 ms

Recommended Value

Any number greater than the minimum value of 1 ms up to 4294967295 ms.

Example

```
SQLNET.RECV_TIMEOUT=10 ms
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.90 SQLNET.SEND_TIMEOUT

Use the sqlnet.ora parameter SQLNET.SEND_TIMEOUT to specify the duration of time in which a database must complete send operations to clients after establishing connections.

Purpose

To specify the time for a database to complete send operations to clients after establishing connections.

You can specify the time in hours, minutes, seconds, or milliseconds by using the hr, min, sec, or ms keyword respectively. If you do not specify a unit of measurement, then the default unit is sec.

Usage Notes

Setting this parameter is recommended for environments in which clients shut down occasionally or unusually.

If the database server cannot complete a send operation in the time specified, then it logs ORA-12608: TNS: Send timeout occurred messages to the sqlnet.log file. Without this parameter, the database server might continue to send responses to clients that are unable to receive data due to a downed computer or a busy state.

You can also set this parameter on the client-side to specify the duration of time in ms, sec, or min, in which client must complete send operations to the database server after connections are established. It accepts different timeouts with or without space between the value and the unit. If you do not specify a unit of measure, then the default unit is sec. Without this parameter, the client might continue to send requests to a database server that is saturated with requests. If you choose to set the value, then set the value initially to a low value and adjust the value according to system and network capacity.

If necessary, then use this parameter with the SQLNET.RECV_TIMEOUT parameter.

Default

None

Minimum Value

`1 ms`

Recommended Value

Any number greater than the minimum value of `1 ms` up to `4294967295 ms`.

Example

```
SQLNET.SEND_TIMEOUT=3 ms
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.91 SQLNET.URI

Use the `sqlnet.ora` parameter `SQLNET.URI` to specify a database client URI mapping on a web server.

Purpose

To specify a database client URI mapping on a web server.

Usage Notes

Use this parameter to customize a URI for mapping the database websocket requests that come into a web server to the back-end database server. Secure websocket handshaking requests are sent with this URI.

Default

`/sqlnet`

Example 5-6 Example

```
sqlnet.uri="/my_uri_prefix/database/"
```

5.2.92 SQLNET.USE_HTTPS_PROXY

Use the `sqlnet.ora` parameter `SQLNET.USE_HTTPS_PROXY` to enable forward HTTP proxy tunneling for client connections.

Purpose

To enable forward HTTP proxy tunneling for client connections.

Usage Notes

If set to `on`, then clients can tunnel secure connections over forward HTTP proxy using the HTTP CONNECT method. This helps access the public cloud database service because it eliminates the requirement to open an outbound port on a client-side firewall.

This parameter is applicable with Oracle Connection Manager on the server side.

Default

on

Example

SQLNET.USE_HTTPS_PROXY=on

5.2.93 SQLNET.WALLET_OVERRIDE

Use the sqlnet.ora parameter SQLNET.WALLET_OVERRIDE to determine whether a client should override strong authentication credentials with the password credential from the stored wallet.

Purpose

To determine whether a client should override strong authentication credentials with the password credential from the stored wallet to log in to a database.

Usage Notes

- When you use wallets for authentication, the database credentials for user name and password are securely stored in an Oracle wallet. The auto-login feature of the wallet is enabled so that the database does not need a password to open the wallet. From the wallet, the database gets the credentials to access the database for the user.

  Oracle has introduced a new auto-login wallet version (7) with Oracle Database 23ai. Version 6 of the Oracle local auto-login wallet is deprecated.

  You can update your local auto-login wallet by modifying it with orapki.

- Wallet use 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 do not need embedded user names and passwords. Risk is reduced because such passwords are no longer exposed, and password management policies are enforced without changing application code whenever user names or passwords change.

  Users connect using the connect /@*database_name* command instead of specifying a user name and password explicitly. This simplifies the maintenance of the scripts and secures the password management for the applications.

- Middle-tier applications create an Oracle Applications wallet during installation to store an application's identity. The password may be randomly generated rather than hardcoded. When an Oracle application accesses the database, it sets appropriate values for SQLNET.AUTHENTICATION_SERVICES and WALLET_LOCATION. The new wallet-based password authentication code uses the password credential in the Oracle Applications wallet to log in to the database.

  The parameter WALLET_LOCATION is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

  For Oracle Database server, Oracle recommends that you use the WALLET_ROOT system parameter instead of using WALLET_LOCATION.

Values

true | false

Examples

```
SQLNET.WALLET_OVERRIDE=true
```

Related Topics

- My Oracle Support Note 340559.1
- *Oracle Database Security Guide*

5.2.94 SSL_ALLOW_WEAK_DN_MATCH

Use the `sqlnet.ora` parameter `SSL_ALLOW_WEAK_DN_MATCH` to allow the earlier weaker distinguished name (DN) matching behavior during server-side certificate validation.

Purpose

The `SSL_SERVER_DN_MATCH` parameter controls the DN matching behavior. DN matching adds another client-side check on both the listener and server certificates to ensure that the certificates are the correct ones that the client expects.

Starting with Oracle Database 23ai, the DN matching behavior is enhanced for better security. You can use the `SSL_ALLOW_WEAK_DN_MATCH` parameter to revert to the earlier DN matching behavior, that is, checking only the server certificate and allowing a service name check for partial DN matching.

Usage Notes

This parameter, introduced with Oracle Database 23ai, provides you with a longer period of time to adjust to the new DN matching behavior of `SSL_SERVER_DN_MATCH`.

The `SSL_ALLOW_WEAK_DN_MATCH` parameter, though new to Oracle Database 23ai, is deprecated and will be removed in a future release. Oracle recommends that you get new certificates or change your DN matching strategy.

Values

- `TRUE` | `ON` | `YES` | `1`:

  Allows `SSL_SERVER_DN_MATCH` to revert to its earlier (pre-Oracle Database release 23ai) DN matching behavior. DN matching only checks the server certificate (but not the listener certificate), and allows to check the service name for partial DN matching.

- `FALSE` | `OFF` | `NO` | `0`:

  Enforces `SSL_SERVER_DN_MATCH` to use the enhanced DN matching behavior. DN matching checks both the listener and server certificates, and does not allow a service name check for partial DN matching.

Default

```
FALSE
```

Example

```
SSL_ALLOW_WEAK_DN_MATCH=FALSE
```

Related Topics

- SSL_SERVER_DN_MATCH
  Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.
- SSL_SERVER_CERT_DN
  Use the `SSL_SERVER_CERT_DN` parameter to specify the distinguished name (DN) of the database server.
- *Oracle Database Security Guide*

5.2.95 SSL_CERTIFICATE_ALIAS

Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_ALIAS` to specify the alias of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.

Purpose

To specify the alias that you have provided when storing the client certificate in an Oracle Database wallet.

When encrypting mTLS connections between the database client and database server, the database client needs to provide a signed certificate to the database server. You can store this client certificate in an Oracle Database wallet or Microsoft Certificate Store (MCS). If there is more than one certificate that can be used, the user or application settings can specify the specific one to connect with. This choice can be made manually by the user via graphical user interface (GUI) or automatically by the application using a thumbprint or alias name. A thumbprint or alias name can uniquely identify the client certificate.

This parameter instructs the client to automatically select a particular certificate using the specified alias name. Thus, the user does not need to manually select the correct client certificate from the list available in a wallet.

Usage Notes

Use this parameter in the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

`orapki` helps you manage certificates and wallets for Oracle Database. To get the alias name value, run the following command:

```
orapki wallet display -wallet <wallet directory> -pwd <wallet password> -
complete
```

Value

Certificate alias name

Default

None

Examples

- In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
    (SECURITY=(SSL_CERTIFICATE_ALIAS=my_cert))
  )
```

- In the Easy Connect string:

```
tcps://salesserver:1521/sales.us.example.com?
SSL_CERTIFICATE_ALIAS=my_cert
```

- In the `sqlnet.ora` file:

```
SSL_CERTIFICATE_ALIAS=my_cert
```

Related Topics

- *Oracle Database Security Guide*

5.2.96 SSL_CERTIFICATE_THUMBPRINT

Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_THUMBPRINT` to specify the thumbprint of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.

Purpose

To specify the thumbprint signature for an X509 certificate. These thumbprints are automatically generated for certificates.

When encrypting mTLS connections between the database client and database server, the database client needs to provide a signed certificate to the database server. You can store this client certificate in an Oracle Database wallet or Microsoft Certificate Store (MCS). If there is more than one certificate that can be used, the user or application settings can specify the specific one to connect with. This choice can be made manually by the user via graphical user interface (GUI) or automatically by the application using a thumbprint or alias name. A thumbprint or alias name can uniquely identify the client certificate.

This parameter instructs the client to automatically select a particular certificate using the specified thumbprint. Thus, the user does not need to manually select the correct client certificate from the list available in a certificate store.

Usage Notes

Use this parameter in the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

You can specify both the SHA-1 and SHA-256 thumbprint information for the client certificate.

`orapki` helps you manage certificates and wallets for Oracle Database. To get the thumbprint value, run the following command:

```
orapki wallet display -wallet <wallet directory> -pwd <wallet password> -
complete
```

Value

SHA-1 or SHA-256 thumbprint of the client certificate, in the *<Algorithm>*:*<Hash>* format

For example:

```
SHA1:1B:11:01:5A:B1:2C:20:B2:12:34:3E:04:7B:83:47:DE:70:2E:4E:11


SHA256:B3:8A:5B:1A:03:63:83:92:2B:5D:E1:53:61:EE:03:94:0A:56:B4:56:41:7E:41:2
4:41:9B:88:EB:C6:1E:11:23
```

or

```
SHA1:1B11015AB12C20B212343E047B8347DE702E4E11


SHA256:B38A5B1A036383922B5DE15361EE03940A56B456417E4124419B88EBC61E1123
```

Default

None

Examples

- In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))

(SECURITY=(SSL_CERTIFICATE_THUMBPRINT=SHA1:1B:11:01:5A:B1:2C:20:B2:12:34:3
E:04:7B:83:47:DE:70:2E:4E:11))
  )


net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))

(SECURITY=(SSL_CERTIFICATE_THUMBPRINT=SHA1:1B11015AB12C20B212343E047B8347D
E702E4E11))
  )
```

- In the Easy Connect string:

```
tcps://salesserver:1521/sales.us.example.com?
SSL_CERTIFICATE_THUMBPRINT=SHA1:1B11015AB12C20B212343E047B8347DE702E
4E11
```

- In the `sqlnet.ora` file:

```
SSL_CERTIFICATE_THUMBPRINT=SHA256:B38A5B1A036383922B5DE15361EE03940A
56B456417E4124419B88EBC61E1123
```

Related Topics

- *Oracle Database Security Guide*

5.2.97 SSL_CERT_REVOCATION

Use the `sqlnet.ora` parameter `SSL_CERT_REVOCATION` to configure revocation checks for certificates.

Purpose

To configure a revocation check for a certificate.

> **See Also:**
>
> *Oracle Database Security Guide*

Default

`none`

Values

- `none` disables certificate revocation status checking. This is the default value.

> **Note:**
>
> Oracle recommends that you do not set the `SSL_CERT_REVOCATION` parameter to `none` because this removes a critical component in certificate-based authentication. Without certificate revocation status checking, you cannot protect against stolen certificates that are used for authentication. Set the `none` value only in cases where mitigating controls safeguard the use of certificates for authentication, such as network access control lists or Oracle Database Vault policies that limit the database connection to trusted clients.

- `requested` to perform certificate revocation if a Certificate Revocation List (CRL) is available. Reject an TLS connection if the certificate is revoked. If no appropriate CRL is found to determine the revocation status of the certificate and the certificate is not revoked, then accept the TLS connection.

- `required` to perform certificate revocation when a certificate is available. If a certificate is revoked and no appropriate CRL is found, then reject the TLS connection. If no appropriate CRL is found to ascertain the revocation status of the certificate and the certificate is not revoked, then accept the TLS connection.

Example

```
SSL_CERT_REVOCATION=required
```

5.2.98 SSL_CRL_FILE

Use the `sqlnet.ora` parameter `SSL_CRL_FILE` to specify the name of the file in which you assemble the certificate revocation list (CRL) for client authentication.

Purpose

To specify the name of the file where you can assemble the CRL for client authentication.

Usage Notes

This file contains the PEM-encoded CRL files, in order of preference. You can use this file alternatively or in addition to the `SSL_CRL_PATH` parameter. This parameter is only valid if `SSL_CERT_REVOCATION` is set to either `requested` or `required`.

Syntax

`SSL_CRL_FILE=`*`certificate_revocation_list_filename`*

Default

None

Example

```
SSL_CRL_FILE=crl.txt
```

Related Topics

- SSL_CERT_REVOCATION
  Use the `sqlnet.ora` parameter `SSL_CERT_REVOCATION` to configure revocation checks for certificates.
- SSL_CRL_PATH
  Use the `sqlnet.ora` parameter `SSL_CRL_PATH` to specify the destination directory of the certificate revocation list (CRL) for client authentication.
- *Oracle Database Security Guide*

5.2.99 SSL_CRL_PATH

Use the `sqlnet.ora` parameter `SSL_CRL_PATH` to specify the destination directory of the certificate revocation list (CRL) for client authentication.

Purpose

To specify the directory path where CRLs are stored.

Usage Notes

This parameter is only valid if you set `SSL_CERT_REVOCATION` to either `requested` or `required`.

Both DER-encoded (binary format) and PEM-encoded (BASE64) CRLs are supported.

If you want to store CRLs in a local file system directory, then you must use the `orapki` utility to rename CRLs in your file system so the system can locate them.

Syntax

`SSL_CRL_PATH=`*`certificate_revocation_list_path`*

Default

None

Example

```
SSL_CRL_PATH=/home/user1/crldir
```

Related Topics

- SSL_CERT_REVOCATION
  Use the `sqlnet.ora` parameter `SSL_CERT_REVOCATION` to configure revocation checks for certificates.
- SSL_CRL_FILE
  Use the `sqlnet.ora` parameter `SSL_CRL_FILE` to specify the name of the file in which you assemble the certificate revocation list (CRL) for client authentication.
- *Oracle Database Security Guide*

5.2.100 SSL_CIPHER_SUITES

Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).

Purpose

To control the combination of authentication, encryption, and data integrity algorithms used by TLS. By default, the strongest protocol and cipher are negotiated between the database client and server. Setting this parameter will override the default behavior. You must use this parameter only if you have internal security controls that dictate the usage of certain protocol versions.

Usage Notes

Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the

latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.

Enclose the `SSL_CIPHER_SUITES` parameter value in parentheses. Otherwise, the cipher suite setting does not parse correctly.

Default

None

Values

Approved ciphers compatible with TLS 1.3:

- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256` (non-FIPS only)
- `TLS_AES_128_CCM_SHA256`
- `TLS_AES_128_GCM_SHA256`

Approved ciphers compatible with TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

Deprecated ciphers compatible with TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Examples

```
SSL_CIPHER_SUITES=(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

```
SSL_CIPHER_SUITES=(TLS_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

Related Topics

- Set the TLS Cipher Suites on the Server
- Set the TLS Cipher Suites on the Client

5.2.101 SSL_CLIENT_AUTHENTICATION

Use the SSL_CLIENT_AUTHENTICATION parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).

Purpose

To enable client authentication in a TLS connection. The connection can be one-way or two-way (mutual TLS or mTLS).

Usage Notes

When set to TRUE, a two-way TLS connection is initiated. Both the client and server (including the listener) authenticate each other. For example, if you set this parameter to TRUE in the server configuration (server-side sqlnet.ora), then the server attempts to authenticate the client. If you set it to TRUE in the listener configuration (listener.ora), then the listener attempts to authenticate the client.

When set to FALSE, only the client authenticates the server and listener as a one-way TLS connection. For example, if you set this parameter to FALSE in the server configuration, then the server does not authenticate the client. If you set it to FALSE in the listener configuration, then the listener does not authenticate the client.

When set to OPTIONAL, the server behaves as follows:

- If the client sends a certificate, then the connection is completed as a two-way TLS connection after authenticating the client.
- If the client does not send a certificate, then the connection is completed as a one-way TLS connection.

Ensure that this parameter setting is consistent for the server or listener (on one side) and the client (on the other). Otherwise, the connection may fail. For example, if you enable client authentication in the server or listener configuration, then you must enable it in the client configuration.

Default

TRUE

Values

- `TRUE` | `ON` | `YES` | `1`: To enable mTLS
- `FALSE` | `OFF` | `NO` | `0`: To enable one-way TLS
- `OPTIONAL`: To enable both TLS and mTLS

Example

```
SSL_CLIENT_AUTHENTICATION=FALSE
```

Related Topics

- *Oracle Database Security Guide*

5.2.102 SSL_ENABLE_WEAK_CIPHERS

Use the `sqlnet.ora` parameter `SSL_ENABLE_WEAK_CIPHERS` to enable the use of weak Transport Layer Security (TLS) cipher suites.

Purpose

To enable the use of weak TLS ciphers for backward compatibility. You can set this parameter on both the database server and client.

Usage Notes

By default, this parameter is set to `FALSE` to block the use of weak ciphers. This simplifies the passing of compliance audits and improves the overall security of your database. If you want to enable the use of weak ciphers, then set this parameter to `TRUE`.

When set to `FALSE`, you can use only the following strong ciphers:

- `TLS_AES_128_CCM_SHA256`
- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

With the `SSL_ENABLE_WEAK_CIPHERS=FALSE` setting, if you try to use a weak cipher, then the following error messages appear:

- On the database server: `ORA-28860: Fatal SSL error`
- On the database client: `ORA-29039: There are no matching cipher suites`

Values

- `TRUE` | `ON` | `YES` | `1`: To enable weak ciphers
- `FALSE` | `OFF` | `NO` | `0`: To disable weak ciphers

Default

`FALSE`

Example

```
SSL_ENABLE_WEAK_CIPHERS=FALSE
```

Related Topics

- SSL_CIPHER_SUITES
  Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).
- *Oracle Database Security Guide*

5.2.103 SSL_EXTENDED_KEY_USAGE

Use the `sqlnet.ora` parameter `SSL_EXTENDED_KEY_USAGE` to specify the purpose certificate keys.

Purpose

To specify the purpose of the key in a certificate.

Usage Notes

When you specify this parameter, Oracle uses the certificate with the matching extended key.

Values

`client authentication`

Example

```
SSL_EXTENDED_KEY_USAGE="client authentication"
```

5.2.104 SSL_SERVER_DN_MATCH

Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.

Purpose

To enforce server-side certificate validation through DN matching.

The purpose of adding this DN matching parameter for the client is to further improve security on a Transport Layer Security (TLS) connection. A TLS connection relies on the client to verify if the database server certificate is valid and signed by a trusted root

certificate. The listener and server certificate DN matching adds another client-side check on the listener and server certificates to ensure that the certificates are the correct ones that the client expects.

Usage Notes

- If you set this parameter to `TRUE`, then in addition to verifying the server's certificate chain, the client enforces another check against the listener and server through DN matching.
- You can configure either partial DN matching or full DN matching.

  Through partial DN matching, the client checks the `HOSTNAME` parameter (in the client `sqlnet.ora` file or connect string) against a host name in the certificate DN or certificate Subject Alternative Name (SAN) field. The client checks `HOSTNAME` against both the listener and server certificates in this order:

  1. The client first compares `HOSTNAME` with a host name in the listener certificate's DN. For example, CN part of DN:

     ```
     "c=us,o=examplecorporation,cn=sales.us.example.com"
     ```

  2. If no match is found in the listener certificate's DN, then the client compares `HOSTNAME` with a host name in the listener certificate's SAN field. For example:

     ```
     "DNS:sales.us.example.com"
     ```

     If no match is found in the listener certificate's SAN field, then the client does not try connecting to the server and the connection fails.
  3. If the listener certificate check succeeds, then the client performs similar checks on the server certificate. That is, the client first compares `HOSTNAME` with a host name in the server certificate's DN.
  4. If no match is found in the server certificate's DN, then the client compares `HOSTNAME` with a host name in the server certificate's SAN field.

  Through full DN matching, the client checks the complete DN in `SSL_SERVER_CERT_DN` against the certificate DN of both the listener and server certificates. To enforce a full DN match, specify the complete DN using the `SSL_SERVER_CERT_DN` parameter in the `tnsnames.ora` file or connect string.
- Oracle recommends that you use the same certificate for both the listener and server.

  If you use different certificates with different DNs for the listener and server, then full DN matching fails. In this case, you need to get new certificates with the same DN (for full DN matching) or you need to change your DN matching strategy. If you have configured partial DN matching, then it may also fail if `HOSTNAME` is not found in the certificate DN or SAN fields of both the listener and server certificates.
- Prior to Oracle Database 23ai, partial DN matching checked against host name and SAN only in the server certificate. If a match was not found, then along with the host name and SAN, it also checked the `SERVICE_NAME` parameter. Similarly, full DN matching checked against the complete DN only in the server certificate.

  If you want to revert to the earlier weaker DN matching behavior (that is, checking only the server certificate and allowing a service name check for partial DN matching), then set `SSL_ALLOW_WEAK_DN_MATCH=TRUE`. However, note that the `SSL_ALLOW_WEAK_DN_MATCH` parameter is deprecated and will be removed in a future release. Oracle recommends that you get new certificates or change your DN matching strategy.

Default

NO

Values

- YES | ON | TRUE | 1:

  To enforce partial or full DN matching. If the DN matches the host name or SAN in both the listener and server certificates, then the connection succeeds. If the DN does not match the host name or SAN in the server or listener certificate, then the connection fails.

- NO | OFF | FALSE | 0:

  To not enforce DN matching. If the DN does not match the host name or SAN in the sever or listener certificate, then the connection is successful, but an error is logged to the sqlnet.log file.

Example

```
SSL_SERVER_DN_MATCH=YES
```

Related Topics

- SSL_SERVER_CERT_DN
  Use the SSL_SERVER_CERT_DN parameter to specify the distinguished name (DN) of the database server.
- SSL_ALLOW_WEAK_DN_MATCH
  Use the sqlnet.ora parameter SSL_ALLOW_WEAK_DN_MATCH to allow the earlier weaker distinguished name (DN) matching behavior during server-side certificate validation.
- *Oracle Database Security Guide*

5.2.105 SSL_VERSION

Use the SSL_VERSION parameter to define valid Transport Layer Security (TLS) versions to be used for connections.

Purpose

To define the version of TLS that must run on the systems with which the database server communicates. By default, the database server and client negotiate the strongest security protocol. Oracle does not recommend modifying this parameter, unless your security requirements mandate the usage of certain protocol versions.

Usage Notes

- Clients, listeners, and database servers must use compatible versions. Modify this parameter only when necessary to enforce the use of the more secure TLS protocol and not allow clients that only work with the older TLS protocols. The current default uses TLS 1.3, which is the version required for multiple security compliance requirements. If you need to specify TLS 1.2, then also include TLS 1.3 to allow more secure connections.
- In addition to sqlnet.ora, listener.ora, and cman.ora, you can specify this parameter under the SECURITY section of tnsnames.ora or directly as part of the

connect string. The parameter value specified in the connect string takes precedence over the other specified values.

- Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

  In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.

- Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.

  SSLv3 is a much less secure protocol to secure the database server-to-client connection. Instead of using SSLv3, allow the database server and client to negotiate the most secure protocol that is common between the server and the client. Oracle Database 23ai provides TLS 1.2 and TLS 1.3 protocols for certificate-based network encryption.

- If you set `SSL_VERSION` to `undetermined`, then the most secure TLS protocol version is used. You can also use the `SSL_VERSION=undetermined` setting in the connect string for a specific connection to override the `SSL_VERSION` value configured in the `sqlnet.ora`, `listener.ora`, or `cman.ora` file.

- If you do not set `SSL_VERSION` to any value, then all the supported TLS protocol versions are tried starting with the most secure version. This is typically the most common configuration, ensuring that the strongest protocol is chosen during TLS negotiation.

Values

`undetermined` | `TLSv1.2` | `TLSv1.3`

Default

`undetermined`

Syntax and Examples

- To specify a single protocol version:

  ```
  SSL_VERSION=TLS_protocol_version
  ```

  For example:

  ```
  SSL_VERSION=TLSv1.3
  ```

- To specify multiple protocol versions, use a comma-separated string of values, enclosed in parenthesis:

  ```
  SSL_VERSION=(TLS_protocol_version1,TLS_protocol_version2)
  ```

For example:

```
SSL_VERSION=(TLSv1.2,TLSv1.3)
```

> **Note:**
>
> Do not enclose protocol versions in parenthesis while specifying this parameter in the `tnsnames.ora` file or as part of the connect string, otherwise the setting will not parse correctly. For example:
>
> ```
> net_service_name=
>   (DESCRIPTION=
>      (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver)(PORT=1522))
>      (SECURITY=(SSL_VERSION=TLSv1.2,TLSv1.3))
>   )
> ```

Related Topics

- Set the Required TLS Version on the Server
- Set the Required TLS Version on the Client

5.2.106 TCP.ALLOWED_PROXIES

Use the `sqlnet.ora` parameter `TCP.ALLOWED_PROXIES` to specify a list of the Oracle Connection Manager (CMAN) addresses that can forward client IP address to the database server.

Purpose

To specify a list of the CMAN addresses (IP addresses or host names) that can forward client IP address to the database server.

Usage Notes

Use this parameter in the server-side `sqlnet.ora` file to list the allowed CMAN instances.

In addition to the `TCP.ALLOWED_PROXIES` parameter, you must set the `ENABLE_IP_FORWARDING` parameter in the `cman.ora` file to enable client address forwarding. CMAN will forward client address only if `ENABLE_IP_FORWARDING` is set to `ON`.

You can use the `SYS_CONTEXT ('USERENV','IP_ADDRESS')` function to query the forwarded client address details.

Default

None

Value

A comma-separated list of IP addresses or host names from which you want to allow client address forwarding.

Example

```
TCP.ALLOWED_PROXIES=(10.1.1.1/24,cmanhost1.example.com)
```

Related Topics

- ENABLE_IP_FORWARDING
  Use the `cman.ora` parameter `ENABLE_IP_FORWARDING` to forward client IP address to the database server.
- *Oracle Database SQL Language Reference*

5.2.107 TCP.CONNECT_TIMEOUT

Use the `sqlnet.ora` parameter `TCP.CONNECT_TIMEOUT` to specify the amount of time in which a client must establish TCP connections to database servers.

Purpose

To specify the time in `ms`, `sec`, or `min`, for a client to establish a TCP connection (`PROTOCOL=tcp` in the TNS connect address) to the database server.

Usage Notes

If a TCP connection to the database is not established in the specified amount of time, then the connection attempt ends. The client receives the following error:

`ORA-12170: Cannot connect. TCP connect timeout of` *`time_interval`* `for` *`host_port or key`*`. (CONNECTION_ID=`*`ID_string`*`).`

The timeout applies to each IP address that resolves to a host name. It accepts different timeouts with or without space between the value and the unit. For example, if a host name resolves to an IPv6 and an IPv4 address, and if the host is not reachable through the network, then the connection request times out twice because there are two IP addresses. In this example, the default timeout setting of `60` causes a timeout in `120 seconds`. If you do not specify a unit of measure, then the default unit is `sec`.

Default

60

Example

```
TCP.CONNECT_TIMEOUT=10 ms
```

5.2.108 TCP.EXCLUDED_NODES

Use the `sqlnet.ora` parameter `TCP.EXCLUDED_NODES` to specify which clients are denied access to the database.

Purpose

To specify which clients are denied access to the database.

Usage Notes

This parameter is only valid when you set the TCP.VALIDNODE_CHECKING parameter to `yes`.

You can use wildcards in this parameter for IPv4 addresses and CIDR notation for IPv4 and IPv6 addresses.

Syntax

```
TCP.EXCLUDED_NODES=(hostname | ip_address, hostname | ip_address, ...)
```

Example

```
TCP.EXCLUDED_NODES=(finance.us.example.com, mktg.us.example.com, 192.0.2.25,
 172.30.*, 2001:DB8:200C:417A/32)
```

5.2.109 TCP.INVITED_NODES

Use the `sqlnet.ora` parameter `TCP.INVITED_NODES` to specify which clients are allowed access to the database.

Purpose

To specify which clients are allowed access to the database. This list takes precedence over the `TCP.EXCLUDED_NODES` parameter if both lists are present.

Syntax

```
TCP.INVITED_NODES=(hostname | ip_address, hostname | ip_address, ...)
```

Usage Notes

- This parameter is only valid when you set the TCP.VALIDNODE_CHECKING parameter to `yes`.
- This parameter accepts wildcards for IPv4 addresses and CIDR notation for IPv4 and IPv6 addresses.

Example

```
TCP.INVITED_NODES=(sales.us.example.com, hr.us.example.com, 192.0.*,
 2001:DB8:200C:433B/32)
```

5.2.110 TCP.NODELAY

Use the `sqlnet.ora` parameter `TCP.NODELAY` to preempt delays in buffer flushing within the TCP/IP protocol stack.

Purpose

To preempt delays in buffer flushing within the TCP/IP protocol stack.

Default

yes

Values

`yes | no`

Example

```
TCP.NODELAY=yes
```

5.2.111 TCP.QUEUESIZE

Use the sqlnet.ora parameter TCP.QUEUESIZE to configure the maximum length of queues for pending connections on TCP listening sockets.

Purpose

To configure the maximum length of the queue for pending connections on a TCP listening socket.

Default

System-defined maximum value. The defined maximum value for Linux is 128.

Values

Any integer value up to the system-defined maximum.

Examples

```
TCP.QUEUESIZE=100
```

5.2.112 TCP.VALIDNODE_CHECKING

Use the `sqlnet.ora` parameter `TCP.VALIDNODE_CHECKING` to enable and disable valid node checking for incoming connections.

Purpose

To enable and disable valid node checking for incoming connections.

Usage Notes

If you set this parameter to `yes`, then incoming connections are allowed only if the connections originate from a node that conforms to a list that you specified in the TCP.INVITED_NODES or TCP.EXCLUDED_NODES parameters.

The TCP.INVITED_NODES and TCP.EXCLUDED_NODES parameters are valid only when you set the TCP.VALIDNODE_CHECKING parameter to `yes`.

You must set this parameter and the dependent parameters, TCP.INVITED_NODES and TCP.EXCLUDED_NODES, in the `sqlnet.ora` file of the listener. This is important in Oracle RAC environments where listeners run from the Oracle Grid Infrastructure home. Setting the parameter in the database home does not have an effect in Oracle RAC environments. In such environments, you must include the address of all Single Client Access Name (SCANs), Virtual IPs (VIPs), local IP in the TCP.INVITED_NODES list.

In VLAN environments, the `sqlnet.ora` file present in the Oracle Grid Infrastructure homes should include all of the addresses of all of the VLANs. The VLANs perform the network segregation, whereas the values that are set for `INVITED_NODES` enables or restricts access to databases within the VLANs.

If multiple databases within the same VLAN require different `INVITED_NODE` lists, then you must configure separate listeners.

Default

no

Values

`yes | no`

Example

```
TCP.VALIDNODE_CHECKING=yes
```

5.2.113 TENANT_ID

Use the `TENANT_ID` parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.

Purpose

To specify the ID of the Azure AD tenant in which your Azure AD application is registered. This is the unique tenant ID that identifies your database instance in Azure AD.

Usage Notes

You use this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` token-based authentication flows.

This is an optional parameter. If you have configured the Azure SDKs, then the client driver automatically searches for the tenant ID in the SDK configuration. If you have not configured the SDKs, then you must set this parameter (along with other required

parameters, such as CLIENT_ID and CLIENT_CERTIFICATE). Otherwise, an error message appears prompting you to configure all required parameters.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or tnsnames.ora connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the sqlnet.ora file, Easy Connect syntax, or tnsnames.ora connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the tenant ID value by logging in to the Azure portal. This is listed as Tenant ID on the Tenant Properties page.

Examples

In the tnsnames.ora file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123)
          (REDIRECT_URI=http://localhost:1575))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the sqlnet.ora file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3
TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123
REDIRECT_URI=http://localhost:1575
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-
a12bcdab0123&REDIRECT_URI=http://localhost:1575
```

In these examples, the optional CLIENT_ID parameter is not specified. Thus, the client automatically gets the client ID value from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

5.2.114 TNSPING.TRACE_DIRECTORY

Use the sqlnet.ora parameter TNSPING.TRACE_DIRECTORY to specify the destination directory for the TNSPING utility trace file, tnsping.trc.

Purpose

To specify the destination directory for the TNSPING utility trace file, tnsping.trc.

Default

The ORACLE_HOME/network/trace directory.

Example

```
TNSPING.TRACE_DIRECTORY=/oracle/traces
```

5.2.115 TNSPING.TRACE_LEVEL

Use the sqlnet.ora parameter TNSPING.TRACE_LEVEL to enable or disable TNSPING utility tracing at a specified level.

Purpose

To enable or diable TNSPING utility tracing at a specified level.

Default

off

Values

- off for no trace output
- user for user trace information
- admin for administration trace information
- support for Oracle Support Services trace information

Example

```
TNSPING.TRACE_LEVEL=admin
```

5.2.116 TOKEN_AUTH

Use the TOKEN_AUTH parameter to configure token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) or Microsoft Azure

Active Directory (Azure AD) users. With this setting, the database client looks for a token file when a / (slash) login is used.

Purpose

Token-based access enforces strong authentication, which enables a more secure access to the database. IAM users can connect to OCI Database as a Service (DBaaS) databases, and Azure AD users can connect to Oracle Databases (cloud or on-premises).

Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Usage Notes for IAM

- An OCI IAM token (`db-token`), which is obtained from IAM using Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI) or programmatically from the OCI Software Development Kit (SDK), is a proof-of-possession (PoP) token with an expiration time and scope.

  You can use one of the IAM user credentials, such as API-key, security token, resource principal, instance principal, or delegation token to retrieve the `db-token` and private key from IAM.
- These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.
- You must configure the TCPS protocol (`PROTOCOL=tcps`) and set the `SSL_SERVER_DN_MATCH` parameter to `TRUE` for token-based authentication.
- When an IAM user logs in using `/@`*`connect_identifier`* (and `TOKEN_AUTH` is set to `OCI_TOKEN`), the `TOKEN_AUTH=OCI_TOKEN` setting along with `/@`*`connect_identifier`* instructs the database client to get the `db-token` and private key from either the default directory or the location specified by `TOKEN_LOCATION`.
- If your client application is updated to retrieve tokens from IAM, then you can override the `TOKEN_AUTH=OCI_TOKEN` setting. The client application gets the `db-token` and private key from IAM and sends as attributes to the database client using the client API. In this case, you do not need to specify the `TOKEN_AUTH` and `TOKEN_LOCATION` parameters.
- The general IAM token-based authentication process is as follows:

  1. An IAM user or application in OCI first requests the `db-token` from IAM by using API-key, security token, resource principal, service principal, instance principal, or delegation token (delegation token is available only in the Cloud Shell).

     To use a security token, you need to generate it by completing the browser authentication process and then request the `db-token` using that security token. If the IAM policy that authorizes you to be issued the `db-token` exists, then the `db-token` is returned.

     You request the `db-token` using OCI CLI (or OCI SDK for applications). For example, run the following OCI CLI command to request the `db-token` by using an API-key (`apikey`):

     ```
     $ oci iam db-token get --profile scott
     ```

The `profile` option specifies the profile for which you want to access the IAM user credentials and retrieve the `db-token`.

For more information on using OCI CLI, see the `get` command details in *Oracle Cloud Infrastructure CLI Command Reference*.

2. OCI CLI references the `config` file (that stores your IAM user credentials as part of the profile) and makes a call to IAM to get the `db-token`. The `db-token` and private key files are written at the default or specified token location.
3. You can specify the `TOKEN_LOCATION` parameter to override the default directory where the `db-token` and private key files are stored.

   The database client gets the `db-token` and private key from the default token location or the location specified by `TOKEN_LOCATION`, signs the `db-token` with the private key and sends it to the database server. The database server verifies the `db-token` and gets the group membership information for the user. If the IAM user is mapped to a database schema (exclusively or shared), then the login is completed.

- The following authentication flows enable the database client to directly retrieve the `db-token` with IAM Single-Sign On (SSO) credentials.

  Note that this feature is available in environments that use JDBC-thin clients, ODP.NET Core classes, or ODP.NET Managed Driver classes. For JDBC-thin clients, you can set this in the `tnsnames.ora` or Easy Connect connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can set this in the `sqlnet.ora`, `tnsnames.ora`, or Easy Connect connect string. The parameter value specified in the connect string takes precedence. To configure this feature for JDBC-thin clients, see *Oracle Database JDBC Developer's Guide* and for ODP.NET, see *Oracle Data Provider for .NET Developer's Guide*.

  – `TOKEN_AUTH=OCI_INTERACTIVE` specifies the OCI Interactive flow. This authenticates the token request interactively using a web browser, and is useful for client-side web applications or desktop applications.

    The database client gets a default profile (named `DEFAULT`) from the OCI configuration file, which is stored either in the default directory or at the location specified by the `OCI_CONFIG_FILE` parameter. After validating the user's region against a list of valid regions, the client launches an authentication request to the user in a web browser, prompting to log in using the IAM user name and password along with any additional factors required by IAM.

    Optionally, you can override the `DEFAULT` profile set in the configuration file by specifying the `OCI_PROFILE` parameter.

  – `TOKEN_AUTH=OCI_API_KEY` specifies the OCI API Key flow. This authenticates the token request with IAM using an IAM-recognized API-key.

    The database client reads the file system location of the API-key from the user's `DEFAULT` profile in the OCI configuration file, from either the default configuration file directory or the location specified by `OCI_CONFIG_FILE`.

    Optionally, you can override the user's `DEFAULT` profile set in the configuration file by specifying the `OCI_PROFILE` parameter.

  – `TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL` specifies the OCI Instance Principal flow. This authenticates the token request with IAM as an OCI instance principal for applications running on OCI compute instances.

- `TOKEN_AUTH=OCI_DELEGATION_TOKEN` specifies the OCI Delegation Token flow. This authenticates the token request with IAM using a delegation token for applications running in an OCI Cloud Shell.
- `TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL` specifies the OCI Resource Principal flow. This authenticates the token request with IAM as an OCI resource principal for applications running in a container (as an OCI function).
- `TOKEN_AUTH=OCI_DEFAULT` specifies the Default flow. With this setting, the client driver reads the predefined environment variables from the SDK configuration, evaluates each authentication flow in a sequence, and then assigns the most appropriate flow based on the environment where the application is running.

  This is the sequence in which the driver evaluates each authentication flow with `OCI_DEFAULT`:

  1. OCI API Key: The driver first checks if a configuration file is present at the location specified by the `OCI_CONFIG_FILE` parameter or at the default location (`$HOME/.oci/config`). The driver then checks if the file contains a profile matching the name configured by the `OCI_PROFILE` parameter or the default name (`DEFAULT`). Finally, the driver checks if the profile is configured with an entry named `key_file`. If all of these checks succeed, then authentication with an API key is used. If any of these checks fail, then the driver proceeds to the next step.
  2. OCI Delegation Token: The driver first checks if the `OCI_CONFIG_FILE` environment variable is set. The driver then checks if a file is present at the location configured by the `OCI_CONFIG_FILE` environment variable. The driver then checks if the file contains a profile named `DEFAULT`. Finally, the driver checks if the profile is configured with an entry named `delegation_token_file`. If all of these checks succeed, then authentication with a delegation token is used. If any of these checks fail, then the driver proceeds to the next step.
  3. OCI Resource Principal: The driver first checks if the `OCI_RESOURCE_PRINCIPAL_VERSION` environment variable is set. The driver then checks if the variable is set to version `2.2` or `1.1`. If the variable is set to `2.2`, the driver then checks if the `OCI_RESOURCE_PRINCIPAL_PRIVATE_PEM`, `OCI_RESOURCE_PRINCIPAL_RPST`, and `OCI_RESOURCE_PRINCIPAL_REGION` environment variables are also set. Otherwise, if the variable is set to `1.1`, then the driver checks if the `OCI_RESOURCE_PRINCIPAL_RPT_ENDPOINT` environment variable is also set. If the required variables for a version are set, then authentication as a resource principal is used. If any variable is not set, then the driver proceeds to the next step.
  4. OCI Instance Principal: The driver requests a certificate from the instance metadata service. The base URL of the service is `http://169.254.169.254/opc/v2/`. However, a fallback URL of `http://169.254.169.254/opc/v1/` is used if the `v2` service request fails. If a request to the `v2` or `v1` service succeeds, then authentication as an instance principal is used. If the request fails, then the driver proceeds to the next step.
  5. The driver reports an error indicating that authentication is not possible using any of the authentication flows.

You also need to specify the `OCI_DATABASE` and `OCI_COMPARTMENT` parameters for all these authentication flows, if the OCI database token policy limits you to access only a particular database or databases within a compartment.

> **Note:**
>
> You can also use another IAM credential, IAM database password, to request the `db-token` from IAM. This `db-token` is a bearer token and does not come with a private key. You can configure the database client to request this token using your IAM user name and IAM database password. An application cannot pass this type of `db-token` to the client. In this case, you use a different parameter setting (`PASSWORD_AUTH=OCI_TOKEN`).
>
> Unlike the API-key, security token, resource principal, service principal, instance principal, and delegation token that require an application or tool to get a token, the IAM database password can only be used by the database client to retrieve the token. See PASSWORD_AUTH.

Default Setting for IAM

None

Table 5-2 Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_TOKEN`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Or`
	`acleContext")`
	`(TOKEN_AUTH=OCI_TOKEN))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.exam`
	`ple.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_TOKEN`
	
	In these examples, the optional `TOKEN_LOCATION` parameter is not specified. Thus, the client automatically gets the `db-token` and private key from the default token location.

Table 5-2 (Cont.) Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_INTERACTIVE`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Or`
	`acleContext")`
	`(TOKEN_AUTH=OCI_INTERACTIVE))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.exam`
	`ple.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_INTERACTIVE`
	
	In these examples, the optional `OCI_CONFIG` and `OCI_PROFILE` parameters are not specified. Thus, the client automatically gets the `DEFAULT` profile from the default configuration file directory.

Table 5-2 (Cont.) Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_API_KEY`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Or`
	`acleContext")`
	`(TOKEN_AUTH=OCI_API_KEY))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.exam`
	`ple.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_API_KEY`
	
	In these examples, the optional `OCI_CONFIG` and `OCI_PROFILE` parameters are not specified. Thus, the client automatically gets the API-key value from the `DEFAULT` profile stored in the default configuration file directory.

Table 5-2 (Cont.) Values and Examples for IAM

Value	Example
TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL
TOKEN_AUTH=OCI_DELEGATION_TOKEN	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=OCI_DELEGATION_TOKEN)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_DELEGATION_TOKEN

Table 5-2 (Cont.) Values and Examples for IAM

Value	Example
TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL

Usage Notes for Azure AD

- An Azure AD OAuth2 access token is a bearer token with an expiration time and scope. This token follows the OAuth2.0 standard with Azure AD extensions. You can request these tokens from tools and scripts run on Linux, Microsoft PowerShell, or other environments. You can also request these tokens programmatically using the Microsoft SDKs.
- These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.
- You must configure the TCPS protocol (PROTOCOL=tcps) and set the SSL_SERVER_DN_MATCH parameter to TRUE for token-based authentication.
- When an Azure AD user logs in using */@connect_identifier* (and TOKEN_AUTH is set to OAUTH), the TOKEN_AUTH=OAUTH setting instructs the database client to get the access token from the directory location specified by TOKEN_LOCATION if the token file is named token. If the token file name is different from token, then you must use the file name along with the directory location while specifying the TOKEN_LOCATION parameter.

  The TOKEN_LOCATION parameter is mandatory for Azure AD token-based authentication. The database client gets the token from this location and sends it to the database server.

- If your client application is updated to retrieve tokens from Azure AD, then you can override the `TOKEN_AUTH=OAUTH` setting. Azure AD directly passes the `db-token` as an attribute to the database client using the client API. When the client receives this request, the client sends it to the database server.

  In this case, you do no need to specify the `TOKEN_AUTH` and `TOKEN_LOCATION` parameters.

- The general Azure AD token-based authentication process is as follows:

  1. An Azure AD user or application first requests the access token from Azure AD using one of the supported Microsoft Azure AD authentication flows (resource owner password credentials, authorization code, on-behalf-of (OBO) flow, or client credentials).

     An Azure AD user can connect using any supported utility to retrieve the token and store it in a local file directory.

     You can request the token from tools and scripts run on Linux, Microsoft PowerShell, or other environments. You can also request programmatically using the Microsoft SDKs.
     For detailed examples on how to retrieve an Azure AD OAuth2 access token, see *Oracle Database Security Guide*.

  2. The database client then sends the token to the database server. The database server verifies the token (using the Azure AD public key) and extracts various claims from the token, including user name, app roles, and audience. If the Azure AD principal is mapped to a database schema (exclusively or shared), then the login is completed.

- The following authentication flows enable the database client to directly retrieve an access token with Azure AD SSO credentials.

  Note that this feature is available in environments that use JDBC-thin clients, ODP.NET Core classes, or ODP.NET Managed Driver classes. For JDBC-thin clients, you can set this in the `tnsnames.ora` or Easy Connect connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can set this in the `tnsnames.ora` connect string or `sqlnet.ora` file (except `REDIRECT_URI` and `CLIENT_CERTIFICATE`). The parameter value specified in the connect string takes precedence. To configure this feature for JDBC-thin clients, see *Oracle Database JDBC Developer's Guide* and for ODP.NET, see *Oracle Data Provider for .NET Developer's Guide*.

  – `TOKEN_AUTH=AZURE_INTERACTIVE` specifies the Azure OAuth2 Interactive flow. This authenticates the token request interactively using a web browser, and is useful for client-side web applications or desktop applications.

    The database client launches an authentication request to the user (either in a dialog box if the user is using a web application or as a prompt if the user is working in a command line shell), prompting to log in using the Azure AD user name and password. After logging in to the Azure AD account, the user is redirected back to the client application (to its registered redirect URI).

    You must set the `AZURE_DB_APP_ID_URI` and `REDIRECT_URI` parameters to compose the scope URL and redirect URI (or reply URL) for your Azure AD application. The client driver reads the client ID and tenant ID values from the Azure SDK configuration. If required, you can set the `CLIENT_ID` and `TENANT_ID` parameters to override the default values.

  – `TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL` specifies the Azure Service Principal flow. This authenticates the token request as a service principal by using either a client

secret or a client certificate, and is useful for server-side applications (for example, microservices or back-end apps).

If the client driver is not configured with a client secret, then the client driver reads the file system location of the Azure certificate from the `AZURE_CLIENT_CERTIFICATE_PATH` environment variable in the Azure SDK configuration.

You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. The client driver reads the client ID, tenant ID, and client certificate path from the Azure SDK configuration. If required, you can set the `CLIENT_ID`, `TENANT_ID`, and `CLIENT_CERTIFICATE` parameters to override the default values.

- `TOKEN_AUTH=AZURE_MANAGED_IDENTITY` specifies the Azure Managed Identity flow. This authenticates the token request with Azure AD as an Azure managed identity, and is useful for client-side or server-side applications hosted on Azure environments (for example, Azure App Service or Azure virtual machine).

  You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. By default, the client driver uses a system-assigned managed identity from the Azure SDK configuration. If required, you can set the `CLIENT_ID` parameter to configure a user-assigned managed identity for authenticating the token request.

- `TOKEN_AUTH=AZURE_DEVICE_CODE` specifies the Azure Device Code flow. This authenticates the token request interactively, and is useful for client-side applications running on platforms with limited or no browser support (for example, command line tool, such as SQLcl).

  The database client displays a device code and an Azure AD login URL through the standard output of the tool, and prompts the user to enter the device code, Azure AD user name, and Azure AD password on any browser-supporting device (for example, cellphone or laptop). After completing the login in a web browser, the Azure SDK returns an access token to the client. The client sends the access token to the database to authorize the database user session.

  You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. The client driver reads the client ID and tenant ID values from the Azure SDK configuration. If required, you can set the `CLIENT_ID` and `TENANT_ID` parameters to override the default values.

  > **Note:**
  >
  > You must explicitly enable the Azure OAuth2 Interactive and Azure Device Code flows for your Azure AD app in the Azure portal. To do so, on the App registrations - Authentication page, under Advanced Settings, set **Allow public client flows** to **Yes**.

- `TOKEN_AUTH=AZURE_DEFAULT` specifies the Default flow. With this setting, the client driver reads the predefined environment variables from the SDK configuration, evaluates each authentication flow in a sequence, and then assigns the most appropriate flow based on the environment where the application is running.

This is the sequence in which the driver evaluates each authentication flow with `AZURE_DEFAULT`:

1. Azure Service Principal with Client Secret Credentials: The driver checks if client ID and client secret are configured as parameters to the driver or as SDK environment variables. If both are configured, then the driver authenticates as a service principal using a client secret. Otherwise, the driver proceeds to the next step.
2. Azure Service Principal with Client Certificate Credentials: The driver checks if client ID and client certificate are configured as parameters to the driver or SDK environment variables. If both are configured, then the driver authenticates as a service principal using a client certificate. Otherwise, the driver proceeds to the next step.
3. Azure Username Credentials: The driver checks if client ID, username, and password are configured as parameters to the driver or SDK environment variables. If all are configured, then the driver authenticates as a service principal using the username and password. Otherwise, the driver proceeds to the next step.
4. Azure Managed Identity: The driver checks if the `MSI_ENDPOINT` or `IDENTITY_ENDPOINT` environment variable is set. If either is set, then the driver authenticates as a managed identity using the configured endpoint. If neither is set, then the driver checks if the `AZURE_TENANT_ID` and `AZURE_FEDERATED_TOKEN_FILE` environment variables are set. If both are set, then the driver authenticates as a managed identity using the configured token file. If both are not set, then the driver requests an access token from the Azure Instance Metadata Service (IMDS) endpoint. If the request succeeds, then the driver authenticates as a managed identity. Otherwise, the driver proceeds to the next step.
5. Visual Studio Credentials: For ODP.NET Core classes and ODP.NET Managed Driver classes, the driver additionally evaluates the Azure user through Visual Studio Credentials authentication flow. The driver checks if the `TENANT_ID` parameter or the `AZURE_TENANT_ID` environment variable is set and if the Azure user is logged in to Visual Studio. If both the checks succeed, then authentication with the Visual Studio credentials is used. Otherwise, the driver proceeds to the next step.
6. The driver reports an error indicating that authentication is not possible using any of the authentication flows.

Default Setting for Azure AD

None

Table 5-3 Values and Examples for Azure AD

Value	Example

`TOKEN_AUTH=OAUTH`	• If the token file is named `token`, `TOKEN_AUTH=OAUTH`, and `TOKEN_LOCATION="`*`token_file_directory`*`"`:
	In the `tnsnames.ora` file:
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OAUTH)`
	`(TOKEN_LOCATION="/home/dbuser1/access-token"))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	In the `sqlnet.ora` file:
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token"`
	In these examples, the token file name is `token`. Thus, only the directory path (`/home/dbuser1/access-token`) is specified. The client automatically looks for the `token` file in the specified path and gets the access token.
	• If the token file name is different from `token`, `TOKEN_AUTH=OAUTH`, and `TOKEN_LOCATION="`*`token_file_directory/token_filename`*`"`:
	In the `tnsnames.ora` file:
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`

Table 5-3 (Cont.) Values and Examples for Azure AD

Value	Example
	(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=OAUTH) (TOKEN_LOCATION="/home/dbuser1/access-token/mytoken")) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OAUTH TOKEN_LOCATION="/home/dbuser1/access-token/mytoken" In these examples, the token file name is mytoken. Thus, both the file name and directory path (/home/dbuser1/access-token) are specified. The client gets the access token from the mytoken file in the specified path.

Table 5-3 (Cont.) Values and Examples for Azure AD

Value	Example
TOKEN_AUTH=AZURE_INTERACTIVE	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context") (TOKEN_AUTH=AZURE_INTERACTIVE) (AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3) (REDIRECT_URI=http://localhost:1575)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=AZURE_INTERACTIVE AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3 REDIRECT_URI=http://localhost:1575 In these examples, the optional CLIENT_ID and TENANT_ID parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Table 5-3 (Cont.) Values and Examples for Azure AD

Value	Example
TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL) (AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3 In these examples, the optional CLIENT_ID, TENANT_ID, and CLIENT_CERTIFICATE parameters are not specified. Thus, the client automatically gets the required values from the SDK configuration.

Table 5-3 (Cont.) Values and Examples for Azure AD

Value	Example
TOKEN_AUTH=AZURE_MANAGED_IDENTITY	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (TOKEN_AUTH=AZURE_MANAGED_IDENTITY) (AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=AZURE_MANAGED_IDENTITY AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3 In these examples, the optional CLIENT_ID parameter is not specified. The client driver uses a system-assigned managed identity to authenticate the token request with Azure AD.

Table 5-3 (Cont.) Values and Examples for Azure AD

Value	Example
TOKEN_AUTH=AZURE_DEVICE_CODE	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Or acleContext") (TOKEN_AUTH=AZURE_DEVICE_CODE) (AZURE_DB_APP_ID_URI=https:// application.example.com/ 123ab4cd-1a2b-1234-a12b-aa00123b2cd3)) (CONNECT_DATA=(SERVICE_NAME=sales.us.exam ple.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=AZURE_DEVICE_CODE AZURE_DB_APP_ID_URI=https:// application.example.com/ 123ab4cd-1a2b-1234-a12b-aa00123b2cd3 In these examples, the optional CLIENT_ID and TENANT_ID parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Related Topics

- Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
- Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases
- TOKEN_LOCATION
  Use the TOKEN_LOCATION parameter to specify the directory location where token file is stored for token-based authentication.
- OCI_CONFIG_FILE
  Use the OCI_CONFIG_FILE parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.
- OCI_PROFILE
  Use the OCI_PROFILE parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

- OCI_DATABASE
  Use the OCI_DATABASE parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.
- OCI_COMPARTMENT
  Use the OCI_COMPARTMENT parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.
- AZURE_DB_APP_ID_URI
  Use the AZURE_DB_APP_ID_URI parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).
- REDIRECT_URI
  Use the REDIRECT_URI parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.
- CLIENT_ID
  Use the CLIENT_ID parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.
- TENANT_ID
  Use the TENANT_ID parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.
- CLIENT_CERTIFICATE
  Use the CLIENT_CERTIFICATE parameter to specify the file system path to a client certificate that authenticates the database client.

5.2.117 TOKEN_LOCATION

Use the TOKEN_LOCATION parameter to specify the directory location where token file is stored for token-based authentication.

Purpose

To specify the token file directory location. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) or Microsoft Azure Active Directory (Azure AD) users. The database client gets the token from this location and sends it to the database server. For Azure AD, you can also specify the token file name along with the directory location.

Use this parameter along with the TOKEN_AUTH parameter in the tnsnames.ora file, sqlnet.ora file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

Usage Notes for IAM

The TOKEN_LOCATION parameter is optional for IAM token-based authentication. You can use this parameter along with the TOKEN_AUTH parameter to override the default directory where the db-token and private key are stored. This location is used by the database client to retrieve the db-token and private key.

When an IAM user initiates a connection using /@*connect_identifier* (and TOKEN_AUTH is set to OCI_TOKEN), the database client retrieves the db-token and private key from either the default directory or the location specified by TOKEN_LOCATION. The client then signs the db-token using the private key and sends the db-token to the database server.

Default Setting for IAM

- On Linux:

  `/home/username/.oci/db-token`

- On Windows:

  The database client searches for the default directory in this order:

  If the `USERPROFILE` environment variable is set, then the client searches in the `USERPROFILE` directory (for example, `C:\Users\username`).

  If `USERPROFILE` is not set, then the client searches in `HOMEDRIVE` directory (for example, `C:`) with `HOMEPATH` (for example, `\Users\username`).

  For example, the default token location directory on Windows is:

  `C:\Users\username\.oci\db-token`

Values and Examples for IAM

Value	Example

`TOKEN_LOCATION="token_file_directory"`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OCI_TOKEN)`
	`(TOKEN_LOCATION="/home/oracle/.oci/db-token"))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_TOKEN`
	`TOKEN_LOCATION="/home/oracle/.oci/db-token"`

Usage Notes for Azure AD

The `TOKEN_LOCATION` parameter is mandatory for Azure AD token-based authentication. You must use this parameter along with the `TOKEN_AUTH` parameter to specify the directory location where the Azure AD OAuth2 access token is stored. This location is used by the database client to get the access token.

If your token file is named `token`, then specify only the directory path. If the token file name is different from `token`, then you must use the file name along with the directory path.

When an Azure AD user initiates a connection using `/@connect_identifier`, the database client retrieves the access token from the location specified by `TOKEN_LOCATION` and sends the token to the database server.

Default Setting for Azure AD

None

Values and Examples for Azure AD

Value	Example

If the token file is named `token`:
`TOKEN_LOCATION="token_file_directory"`
	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OAUTH)(TOKEN_LOCATION="/home/dbuser1/access-token"))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token"`
	
	In these examples, the token file name is `token`. Thus, only the directory path (`/home/dbuser1/access-token`) is specified. The client automatically looks for the `token` file in the specified path and gets the access token.

Value	Example

If the token file name is different from `token`:
`TOKEN_LOCATION="token_file_directory/token_filename"`
	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=ON)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OAUTH)`
	`(TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"`
	
	In these examples, the token file name is `mytoken`. Thus, both the file name and directory path (`/home/dbuser1/access-token`) are specified. The client gets the access token from the `mytoken` file in the specified path.

Related Topics

- Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
- Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases
- TOKEN_AUTH

5.2.118 USE_CMAN

Use the `sqlnet.ora` parameter `USE_CMAN` to specify client routing to Oracle Connection Manager.

Purpose

To specify client routing to Oracle Connection Manager.

Usage Notes

When set to `true`, the parameter routes the client to a protocol address for Oracle Connection Manager.

When set to `false`, the client picks one of the address lists at random and fails over to the other address list if the chosen `ADDRESS_LIST` fails. With `USE_CMAN=true`, the client always uses the first address list.

If no Oracle Connection Manager addresses are available, then connections are routed through any available listener address.

Default

false

Values

`true | false`

Example

```
USE_CMAN=true
```

5.2.119 USE_DEDICATED_SERVER

Use the sqlnet.ora parameter USE_DEDICATED_SERVER to append `(SERVER=dedicated)` to the `CONNECT_DATA` section of the connect descriptor that the client uses.

Purpose

To append `(SERVER=dedicated)` to the `CONNECT_DATA` section of the connect descriptor used by the client.

Usage Notes

The value for this parameter overrides the current value of the SERVER parameter in the `tnsnames.ora` file.

When set to `on`, the parameter `USE_DEDICATED_SERVER` automatically appends `(SERVER=dedicated)` to the connect data for a connect descriptor. This enables connections from this client use a dedicated server process, even if shared server is configured.

Default

off

Values

- `on` to append `(SERVER=dedicated)`
- `off` to send requests to existing server processes

Example

```
USE_DEDICATED_SERVER=on
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

5.2.120 WALLET_LOCATION

Use the WALLET_LOCATION parameter to specify the location of Oracle wallets.

Purpose

To specify the directory path where you want to create and store an Oracle wallet. Wallets securely contain certificates, secrets, private keys, and trust points used by Oracle Database.

Usage Notes

- Deprecation of the server-side setting:

  The parameter WALLET_LOCATION is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

  For Oracle Database server, Oracle recommends that you use the WALLET_ROOT system parameter instead of using WALLET_LOCATION.

- Where to set this parameter:

  You can set WALLET_LOCATION in the sqlnet.ora file to specify a common wallet location for all connections. You can also set it in the connect string or tnsnames.ora file to specify a different wallet location for a particular connection.

  Use of WALLET_LOCATION in the connect string or tnsnames.ora overrides the sqlnet.ora WALLET_LOCATION setting for the specific tnsnames.ora service. The tnsnames.ora WALLET_LOCATION setting enables a client to initiate multiple TLS sessions using different TLS certificates in the same client process.

- Setting to use the system default certificate store instead of a client-side wallet:

  The Linux and Windows database clients can use the system default certificate store to validate the Oracle Database server certificate, instead of creating a local wallet with root certificate. The default certificate store is located in /etc/pki/tls/cert.pem on Linux and Microsoft Certificate Store (MCS) on Windows.

  If you set WALLET_LOCATION=SYSTEM in the connect string (in tnsnames.ora or directly to the command line), then the database client uses the default certificate store to validate the server certificate. In this case, the server certificate needs to be signed by a trusted root certificate that is already installed in the default certificate store.

  For example:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1234))
       (SECURITY=(WALLET_LOCATION=SYSTEM))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

- Order in which the database client searches for a client wallet:

  1. The database client first tries to use a wallet from the WALLET_LOCATION directory specified in the connect string.
  2. If no wallet is present, then the client searches for the WALLET_LOCATION parameter value in the sqlnet.ora file.

3. If no wallet is present, then the client searches for a wallet in the `$TNS_ADMIN` environment variable directory.
4. If no wallet is present, then the client searches in the default wallet location, that is, `/etc/ORACLE/WALLETS/username` on Linux and `C:\Users\username\\ORACLE\WALLETS` on Windows.
5. If no wallet is present, then the client uses the wallet from the system default certificate store.

You can specify `WALLET_LOCATION` as `SYSTEM` in the connect string to ignore all the wallet configurations and use the system default certificate store instead.

- Setting for walletless TLS connections:
  The `WALLET_LOCATION` parameter is optional for TLS connections that do not use a client wallet. If you do not include `WALLET_LOCATION` in the connect string, `tnsnames.ora`, or `sqlnet.ora`, then the driver automatically picks up common root certificates from the system default certificate store (if the system is Windows or Linux).

  However, you may need to perform additional steps in the following cases:

  – If `WALLET_LOCATION` is set in `sqlnet.ora` for all connections, then you can override this setting for a specific connection that does not need a client wallet (using `WALLET_LOCATION=SYSTEM` in the connect string).
  – If a wallet is present in the `$TNS_ADMIN` environment variable directory, then the database client uses the `$TNS_ADMIN` path as the default wallet location. In this case, you can either override the `WALLET_LOCATION` setting (using `WALLET_LOCATION=SYSTEM` in the connect string) or remove that wallet.

- Storage of wallet files:
  The password-protected wallet is stored in an `ewallet.p12` file. The auto-login and local auto-login wallets are stored in a `cwallet.sso` file.

  For example, if an Oracle wallet is stored in the Microsoft Windows registry and the wallet's key `(KEY)` is `SALESAPP`, then the storage location of the password-protected wallet is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\EWALLET.P12`. The storage location of the auto-login and local auto-login wallets is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\CWALLET.SSO`.

Additional Parameters

Use `SOURCE` to specify the type of storage and storage location for wallets, as follows:

- `METHOD`: Type of storage
- `METHOD_DATA`: Storage location:
  – `DIRECTORY`: Location of wallet on the file system
  – `KEY`: Wallet type and location in the Microsoft Windows registry

Syntax and Examples

The syntax depends on the wallet as follows:

- Wallet on the file system:

```
WALLET_LOCATION=
  (SOURCE=
```

```
   (METHOD=file)
   (METHOD_DATA=
      (DIRECTORY=directory)))
```

For example:

```
WALLET_LOCATION=
  (SOURCE=
      (METHOD=file)
      (METHOD_DATA=
         (DIRECTORY=/etc/oracle/wallets/databases)))
```

- Microsoft certificate store:

```
WALLET_LOCATION=
  (SOURCE=
     (METHOD=mcs))
```

The key-value pair for MCS omits the `METHOD_DATA` parameter because MCS does not use wallets. Instead, Oracle PKI (public key infrastructure) applications obtain certificates, trust points and private keys directly from a user's profile.

- Wallet in the Microsoft Windows registry:

```
WALLET_LOCATION=
   (SOURCE=
      (METHOD=reg)
      (METHOD_DATA=
         (KEY=registry_key)))
```

For example:

```
WALLET_LOCATION=
   (SOURCE=
     (METHOD=reg)
     (METHOD_DATA=
        (KEY=SALESAPP)))
```

Default

None

Related Topics

- *Oracle Database Security Guide*

5.2.121 BEQUEATH_DETACH

Use the `sqlnet.ora` parameter to enable and disable handling signals on Linux and UNIX systems.

Purpose

To enable or disable signal handling on Linux and UNIX systems

Default

no

Values

- `yes` to turn signal handling off

- `no` to leave signal handling on

Example

```
BEQUEATH_DETACH=yes
```

5.3 ADR Diagnostic Parameters in sqlnet.ora

Diagnostic data for critical errors is stored in the `sqlnet.ora` Automatic Diagnostic Repository (ADR).

- About ADR Diagnostic Parameters
  You can use Automatic Diagnostic Repository (ADR) diagnostic parameters when ADR is enabled, which is the default. Oracle ignores non-ADR parameters in the `sqlnet.ora` file when you enable ADR.
- ADR_BASE
  Use the `sqlnet.ora` parameter `ADR_BASE` to specify the base location of the ADR files.
- DIAG_ADR_ENABLED
  Use the `sqlnet.ora` parameter `DIAG_ADR_ENABLED` to enable and disable ADR tracing.
- TRACE_LEVEL_CLIENT
  Use the `sqlnet.ora` parameter `TRACE_LEVEL_CLIENT` to enable and disable client tracing at a specific level.
- TRACE_LEVEL_SERVER
  Use the `sqlnet.ora` parameter `TRACE_LEVEL_SERVER` to enable and disable server tracing at a specific level.
- TRACE_TIMESTAMP_CLIENT
  Use the `sqlnet.ora` parameter `TRACE_TIMESTAMP_CLIENT` to add time stamps to trace events in client trace files.
- TRACE_TIMESTAMP_SERVER
  Use the `sqlnet.ora` parameter `TRACE_TIMESTAMP_CLIENT` to add time stamps to trace events in database trace files.

5.3.1 About ADR Diagnostic Parameters

You can use Automatic Diagnostic Repository (ADR) diagnostic parameters when ADR is enabled, which is the default. Oracle ignores non-ADR parameters in the `sqlnet.ora` file when you enable ADR.

Since Oracle Database 11g, Oracle Database includes an advanced fault diagnostic infrastructure to prevent, detect, diagnose, and resolve problems. The problems might be critical errors such as those that are caused by database code bugs, metadata corruption, or customer data corruption.

When critical errors occur, they are assigned incident numbers. Diagnostic data for the errors, such as traces and dumps, are captured and tagged with the incident number. The data is then stored in ADR, which is a file-based repository outside the database.

The following `sqlnet.ora` parameters are used when you enable ADR (when `DIAG_ADR_ENABLED` is set to `on`):

5.3.2 ADR_BASE

Use the `sqlnet.ora` parameter `ADR_BASE` to specify the base location of the ADR files.

Purpose

To specify the base directory in which Oracle stores tracing and logging incidents when ADR is enabled.

Usage Notes

This parameter is applicable only to clients. On the server side, the ADR base location is defined by the `DIAGNOSTIC_DEST` initialization parameter in the `init.ora` file. See DIAGNOSTIC_DEST in *Oracle Database Reference*.

Default

`ORACLE_BASE` or `ORACLE_HOME/log` (if `ORACLE_BASE` is not defined)

Values

Any valid directory path to a directory with write permission.

Example

```
ADR_BASE=/oracle/network/trace
```

5.3.3 DIAG_ADR_ENABLED

Use the `sqlnet.ora` parameter `DIAG_ADR_ENABLED` to enable and disable ADR tracing.

Purpose

To specify whether ADR tracing is enabled.

Usage Notes

If you set the `DIAG_ADR_ENABLED` parameter to `OFF`, then non-ADR file tracing is used.

Default

`on`

Values

`on | off`

Example 5-7 Example

```
DIAG_ADR_ENABLED=on
```

5.3.4 TRACE_LEVEL_CLIENT

Use the `sqlnet.ora` parameter `TRACE_LEVEL_CLIENT` to enable and disable client tracing at a specific level.

Purpose

To enable client tracing at a specified level or to disable it.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

off or 0

Values

- `off` or `0` for no trace output
- `user` or `4` for user trace information
- `admin` or `10` for administration trace information
- `support` or `16` for Oracle Support Services trace information

Example

```
TRACE_LEVEL_CLIENT=user
```

5.3.5 TRACE_LEVEL_SERVER

Use the `sqlnet.ora` parameter `TRACE_LEVEL_SERVER` to enable and disable server tracing at a specific level.

Purpose

To turn server tracing on at a specified level or to turn it off.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

off or 0

Values

- `off` or `0` for no trace output
- `user` or `4` for user trace information
- `admin` or `10` for administration trace information
- `support` or `16` for Oracle Support Services trace information

Example

```
TRACE_LEVEL_SERVER=admin
```

5.3.6 TRACE_TIMESTAMP_CLIENT

Use the `sqlnet.ora` parameter `TRACE_TIMESTAMP_CLIENT` to add time stamps to trace events in client trace files.

Purpose

To add a time stamp in the form of `dd-mmm-yyyy hh:mm:ss:mil` to every trace event in the client trace file, which has a default name of `sqlnet.trc`.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

on

Values

`on` or `true` | `off` or `false`

Example

```
TRACE_TIMESTAMP_CLIENT=true
```

5.3.7 TRACE_TIMESTAMP_SERVER

Use the `sqlnet.ora` parameter `TRACE_TIMESTAMP_CLIENT` to add time stamps to trace events in database trace files.

Purpose

To add a time stamp in the form of `dd-mmm-yyyy hh:mm:ss:mil` to every trace event in the database server trace file, which has a default name of `svr_pid.trc`.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

on

Values

`on` or `true` | `off` or `false`

Example

```
TRACE_TIMESTAMP_SERVER=true
```

5.4 Non-ADR Diagnostic Parameters in sqlnet.ora Files

Learn about `sqlnet.ora` parameters that you use when you disable ADR.

This section lists the `sqlnet.ora` parameters that are used when you disable ADR.

> **Note:**
>
> The default value of `DIAG_ADR_ENABLED` is `on`. Therefore, the `DIAG_ADR_ENABLED` parameter must explicitly be set to `off` to use non-ADR tracing.

- LOG_DIRECTORY_CLIENT
  Use the `sqlnet.ora` non-ADR diagnostic parameter `LOG_DIRECTORY_CLIENT` to specify the destination directory for client log files.
- LOG_DIRECTORY_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_DIRECTORY_SERVER` to specify the destination directory for database log files.
- LOG_FILE_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_FILE_CLIENT` to specify the name of log files for clients.
- LOG_FILE_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_FILE_SERVER` to specify log file names for the database.
- TRACE_DIRECTORY_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_DIRECTORY_CLIENT` to specify the destination directory for client trace files.
- TRACE_DIRECTORY_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_DIRECTORY_SERVER` to specify the destination directory for database trace files.
- TRACE_FILE_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILE_CLIENT` to specify the names of client trace files.
- TRACE_FILE_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILE_SERVER` to specify the destination directory for database trace output.
- TRACE_FILEAGE_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILEAGE_CLIENT` to specify the maximum age of client trace files in minutes.
- TRACE_FILEAGE_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILEAGE_SERVER` to specify the maximum age of database trace files in minutes.
- TRACE_FILELEN_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILELEN_CLIENT` to specify the size of client trace files in kilobytes.

- TRACE_FILELEN_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILELEN_SERVER` to specify the size of database trace files in kilobytes.
- TRACE_FILENO_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILENO_CLIENT` to specify the number of trace files for client tracing.
- TRACE_FILENO_SERVER
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILENO_SERVER` to specify the number of trace files for database tracing.
- TRACE_UNIQUE_CLIENT
  Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_UNIQUE_CLIENT` to specify whether Oracle creates a unique trace file for each client trace session.

5.4.1 LOG_DIRECTORY_CLIENT

Use the `sqlnet.ora` non-ADR diagnostic parameter `LOG_DIRECTORY_CLIENT` to specify the destination directory for client log files.

Purpose

To specify the destination directory for the client log file.

Usage Notes

Use this parameter when ADR is not enabled.

Default

```
ORACLE_HOME/network/log
```

Values

Any valid directory path.

Example

```
LOG_DIRECTORY_CLIENT=/oracle/network/log
```

5.4.2 LOG_DIRECTORY_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_DIRECTORY_SERVER` to specify the destination directory for database log files.

Purpose

To specify the destination directory for database log files.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace`

Values

Any valid directory path to a directory with write permission.

Example

```
LOG_DIRECTORY_SERVER=/oracle/network/trace
```

5.4.3 LOG_FILE_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_FILE_CLIENT` to specify the name of log files for clients.

Purpose

To specify the name of the log file for the client.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/log/sqlnet.log`

Values

The default value cannot be changed.

5.4.4 LOG_FILE_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `LOG_FILE_SERVER` to specify log file names for the database.

Purpose

To specify the name of the log file for the database.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`sqlnet.log`

Values

Example

```
LOG_FILE_SERVER=svr.log
```

5.4.5 TRACE_DIRECTORY_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_DIRECTORY_CLIENT` to specify the destination directory for client trace files.

Purpose

To specify the destination directory for the client trace file.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace`

Values

Any valid directory path to a directory with write permission.

Example

```
TRACE_DIRECTORY_CLIENT=/oracle/traces
```

5.4.6 TRACE_DIRECTORY_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_DIRECTORY_SERVER` to specify the destination directory for database trace files.

Purpose

To specify the destination directory for the database server trace file. Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace`

Values

Any valid directory path to a directory with write permission.

Example

```
TRACE_DIRECTORY_SERVER=/oracle/traces
```

5.4.7 TRACE_FILE_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILE_CLIENT` to specify the names of client trace files.

Purpose

To specify the name of a client trace file.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace/cli.trc`

Values

Any valid file name.

Example

```
TRACE_FILE_CLIENT=clientsqlnet.trc
```

5.4.8 TRACE_FILE_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILE_SERVER` to specify the destination directory for database trace output.

Purpose

To specify the destination directory for the database server trace output.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace/svr_pid.trc`

Values

Any valid file name. The process identifier (pid) is appended to the name automatically.

Example

```
TRACE_FILE_SERVER=svrsqlnet.trc
```

5.4.9 TRACE_FILEAGE_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILEAGE_CLIENT` to specify the maximum age of client trace files in minutes.

Purpose

To specify the maximum age of client trace files in minutes.

Usage Notes

When the age limit is reached, the trace information is written to the next file. The number of files is specified with the TRACE_FILENO_CLIENT parameter. Use this parameter when ADR is not enabled.

Default

Unlimited

This is the same as setting the parameter to `0`.

Example 5-8 Example

```
TRACE_FILEAGE_CLIENT=60
```

5.4.10 TRACE_FILEAGE_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILEAGE_SERVER` to specify the maximum age of database trace files in minutes.

Purpose

To specify the maximum age of database server trace files in minutes.

Usage Notes

When the age limit is reached, the trace information is written to the next file. The number of files is specified with the TRACE_FILENO_SERVER parameter. Use this parameter when ADR is not enabled.

Default

Unlimited

This is the same as setting the parameter to`0`.

Example 5-9 Example

```
TRACE_FILEAGE_SERVER=60
```

5.4.11 TRACE_FILELEN_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILELEN_CLIENT` to specify the size of client trace files in kilobytes.

Purpose

When the file grows to the specified size, Oracle writes the trace information to the next file. The number of files is specified with the TRACE_FILENO_CLIENT parameter. Use this parameter when ADR is not enabled.

To specify the size of the client trace files in kilobytes (KB).

Usage Notes

Example

```
TRACE_FILELEN_CLIENT=100
```

5.4.12 TRACE_FILELEN_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILELEN_SERVER` to specify the size of database trace files in kilobytes.

Purpose

To specify the size of the database server trace files in kilobytes (KB).

Usage Notes

When the file grows to the specified size, Oracle writes the trace information to the next file. The number of files is specified with the TRACE_FILENO_SERVER parameter. Use this parameter when ADR is not enabled.

Example

```
TRACE_FILELEN_SERVER=100
```

5.4.13 TRACE_FILENO_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILENO_CLIENT` to specify the number of trace files for client tracing.

Purpose

To specify the number of trace files for client tracing.

Usage Notes

When this parameter is set with the TRACE_FILELEN_CLIENT parameter, trace files are used in a cyclical fashion. The first file is filled first, then the second file, and so on. When the last file has been filled, then the first file is re-used, and so on.

When this parameter is set with theTRACE_FILEAGE_CLIENT parameter, trace files are cycled based on their age. The first file is used until the age limit is reached, then the second file is used, and so on. When the last file's age limit is reached, the first file is re-used.

When you set this parameter with both the `TRACE_FILELEN_CLIENT` and `TRACE_FILEAGE_CLIENT` parameters, trace files are replaced when either the size limit or the age limit is reached.

The trace file names are distinguished from one another by their sequence numbers. For example, if the default trace file of `sqlnet.trc` is used, and this parameter is set to 3, then the trace files would be named `sqlnet1.trc`, `sqlnet2.trc` and `sqlnet3.trc`.

In addition, trace events in the trace files are preceded by the sequence number of the file. Use this parameter when ADR is not enabled.

Default

None

Example

```
TRACE_FILENO_CLIENT=3
```

5.4.14 TRACE_FILENO_SERVER

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_FILENO_SERVER` to specify the number of trace files for database tracing.

Purpose

To specify the number of trace files for database server tracing.

Usage Notes

When you set this parameter with the TRACE_FILELEN_SERVER parameter, trace files are used in a cyclical fashion. The first file is filled first, then the second file, and so on. When the last file has been filled, then the first file is re-used.

When you set this parameter with theTRACE_FILEAGE_SERVER parameter, trace files are cycled based on the age of the trace file. The first file is used until the age limit is reached, then the second file is used, and so on. When the last file's age limit is reached, the first file is re-used.

When this parameter is set with both the `TRACE_FILELEN_SERVER` and `TRACE_FILEAGE_SERVER` parameters, trace files are cycled when either the size limit or the age limit is reached.

The trace file names are distinguished from one another by their sequence numbers. For example, if the default trace file of `svr_pid.trc` is used, and this parameter is set to 3, then the trace files would be named `svr1_pid.trc`, `svr2_pid.trc` and `svr3_pid.trc`.

In addition, trace events in the trace files are preceded by the sequence number of the file. Use this parameter when ADR is not enabled.

Default

None

Example

```
TRACE_FILENO_SERVER=3
```

5.4.15 TRACE_UNIQUE_CLIENT

Use the non-ADR diagnostic `sqlnet.ora` parameter `TRACE_UNIQUE_CLIENT` to specify whether Oracle creates a unique trace file for each client trace session.

Purpose

To specify whether a unique trace file is created for each client trace session.

Usage Notes

When you set the value to `on`, a process identifier is appended to the name of each trace file, enabling several files to coexist. For example, trace files named `sqlnetpid.trc` are created if default trace file name `sqlnet.trc` is used. When you set the value to `off`, data from a new client trace session overwrites the existing file. Use this parameter when ADR is not enabled.

Default

on

Values

`on` or `off`

Example

```
TRACE_UNIQUE_CLIENT=on
```

6

Local Naming Parameters in the tnsnames.ora File

This chapter describes the `tnsnames.ora` file configuration parameters.

- Overview of Local Naming Parameters
  This section provides an overview of Oracle Net service names local naming parameters.
- General Syntax of tnsnames.ora
  This section explains the general `tnsnames.ora` file syntax.
- Using Multiple Descriptions in tnsnames.ora Files
  Learn about `tnsnames.ora` file connect descriptors.
- Multiple Address Lists in tnsnames.ora Files
  Learn how to configure multiple address lists in `tnsnames.ora` files.
- Connect-Time Failover and Client Load Balancing with Oracle Connection Managers
  When `tnsnames.ora` file connect descriptors have at least two protocol addresses for Oracle Connection Manager, you can also include parameters for connect-time failover and load balancing in the file.
- Connect Descriptor Descriptions
  Specify connect descriptors using the `DESCRIPTION` parameter. Identify multiple connect descriptors with the `DESCRIPTION_LIST` parameter.
- Protocol Addresses
  Learn about Oracle Net Services protocol address parameters.
- Optional Parameters for Address Lists
  For multiple addresses, you can use the optional parameters to configure address lists.
- Connection Data Section
  Learn how to configure network connections with protocol addresses.
- Security Section
  The security section of the `tnsnames.ora` file specifies these security-related parameters for use with Oracle security features.
- Timeout Parameters
  The timeout section of the `tnsnames.ora` file provides the ability to specify timeout and retry configuration through the TNS connect string.
- Compression Parameters
  The compression section of the `tnsnames.ora` file provides the ability to enable compression and specify compression levels. These parameters can be set at the `DESCRIPTION` level of a connect string.

6.1 Overview of Local Naming Parameters

This section provides an overview of Oracle Net service names local naming parameters.

The `tnsnames.ora` file is a configuration file that contains network service names that are mapped to connect descriptors for the local naming method or net service names that are mapped to listener protocol addresses.

A net service name is an alias that is mapped to a database network address that is contained in a connect descriptor. A connect descriptor contains the location of the listener that is accessed through a protocol address and the service name of the database to which to connect. Clients and database servers that are clients of other database servers use the net service name when connecting with applications.

By default, the `tnsnames.ora` file is located in the *`ORACLE_HOME`*`/network/admin` directory. Oracle Net checks the other directories for the configuration file. For example, the order of checking the `tnsnames.ora` file is as follows:

- The directory specified by the `TNS_ADMIN` environment variable. If the file is not found in the directory specified, then it is assumed that the file does not exist.
- If you do not set the `TNS_ADMIN` environment variable, then Oracle Net first checks the *`ORACLE_BASE_HOME`*`/network/admin` directory.
- If the file is not found in the *`ORACLE_BASE_HOME`*`/network/admin` directory, then Oracle Net checks for the file in the *`ORACLE_HOME`*`/network/admin` directory.

> **Note:**
>
> - On Microsoft Windows, the `TNS_ADMIN` environment variable is used if it is set in the environment of the process. If you do not define the `TNS_ADMIN` environment variable in the environment, or if the process is a service that does not have an environment, then Microsoft Windows scans the registry for a `TNS_ADMIN` parameter.
> - With Oracle Instant Client, the `tnsnames.ora` file is located in the subdirectory of the Oracle Instant Client software. For example, in the `/opt/oracle/instantclient_`*`release_number`*`/network/admin` directory.

6.2 General Syntax of tnsnames.ora

This section explains the general `tnsnames.ora` file syntax.

The basic syntax for a `tnsnames.ora` file is shown in Example 6-1.

Example 6-1 Basic Format of tnsnames.ora File

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS=(protocol_address_information))
   (CONNECT_DATA=
     (SERVICE_NAME=service_name)))
```

In the preceding example, `DESCRIPTION` contains the connect descriptor, `ADDRESS` contains the protocol address, and `CONNECT_DATA` contains database service identification information.

6.3 Using Multiple Descriptions in tnsnames.ora Files

Learn about `tnsnames.ora` file connect descriptors.

A `tnsnames.ora` file can contain net service names with one or more connect descriptors. Each connect descriptor can contain one or more protocol addresses. The following example shows two connect descriptors with multiple addresses. Use the `tnsnames.ora` parameter `DESCRIPTION_LIST` to define the list of connect descriptors.

Example 6-2 Net Service Name with Multiple Connect Descriptors in tnsnames.ora

```
net_service_name=
 (DESCRIPTION_LIST=
  (DESCRIPTION=

   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)))
  (DESCRIPTION=

   (ADDRESS=(PROTOCOL=tcp)(HOST=hr1-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=hr2-svr)(PORT=1521))
   (CONNECT_DATA=
     (SERVICE_NAME=hr.us.example.com))))
```

> **Note:**
>
> Oracle Net Manager does not support multiple connect descriptors for a net service name if you use Oracle Connection Manager.

6.4 Multiple Address Lists in tnsnames.ora Files

Learn how to configure multiple address lists in `tnsnames.ora` files.

The `tnsnames.ora` file supports connect descriptors with multiple lists of addresses, each with its own characteristics. The following example shows two address lists. The first address list features client load balancing and no connect-time failover. These setting apply only to protocol addresses that are within its `ADDRESS_LIST`. The second protocol address list does not enable client load loading balancing, but the list does enable connect-time failover. These settings affect only protocol addresses that are included in its `ADDRESS_LIST`. The client first tries the first or second protocol address at random, then it tries protocol addresses number three and four, in that order, and so on.

Example 6-3 Multiple Address Lists in tnsnames.ora Files

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (LOAD_BALANCE=on)
   (FAILOVER=off)
   (ADDRESS=(protocol_address_information))
   (ADDRESS=(protocol_address_information)))
```

```
(ADDRESS_LIST=
 (LOAD_BALANCE=off)
 (FAILOVER=on)
 (ADDRESS=(protocol_address_information))
 (ADDRESS=(protocol_address_information)))
(CONNECT_DATA=
 (SERVICE_NAME=service_name)))
```

Note:

- Oracle Net Manager supports only the creation of one protocol address list for a connect descriptor.
- Oracle Net Services supports the IFILE parameter in the `tnsnames.ora` file, with up to three levels of nesting. You must add the parameter manually to the file. The following is an example of the syntax:

```
IFILE=/tmp/listener_em.ora
IFILE=/tmp/listener_cust1.ora
IFILE=/tmp/listener_cust2.ora
```

6.5 Connect-Time Failover and Client Load Balancing with Oracle Connection Managers

When `tnsnames.ora` file connect descriptors have at least two protocol addresses for Oracle Connection Manager, you can also include parameters for connect-time failover and load balancing in the file.

Example 6-4 Multiple Oracle Connection Manager Addresses in tnsnames.ora

This example illustrates the failover of multiple Oracle Connection Manager protocol addresses.

```
sample1=
 (DESCRIPTION=
   (SOURCE_ROUTE=yes)
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=host1)(PORT=1630))     # 1
     (ADDRESS_LIST=
       (FAILOVER=on)
       (LOAD_BALANCE=off)                                #  2
       (ADDRESS=(PROTOCOL=tcp)(HOST=host2a)(PORT=1630))
       (ADDRESS=(PROTOCOL=tcp)(HOST=host2b)(PORT=1630)))
     (ADDRESS=(PROTOCOL=tcp)(HOST=host3)(PORT=1521)))     #  3
   (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

The previous syntax does the following:

1. The client connects to the protocol address of the first Oracle Connection Manager as indicated by:

```
(ADDRESS=(PROTOCOL=tcp)(HOST=host1)(PORT=1630))
```

2. Oracle Connection Manager connects to the first protocol address of another Oracle Connection Manager. If the first protocol address fails, then it tries to connect to the second protocol address. This sequence is specified with the following configuration:

```
(ADDRESS_LIST=
  (FAILOVER=on)
  (LOAD_BALANCE=off)
  (ADDRESS=(PROTOCOL=tcp)(HOST=host2a)(PORT=1630))
  (ADDRESS=(PROTOCOL=tcp)(HOST=host2b)(PORT=1630)))
```

3. Oracle Connection Manager connects to the database service using the following protocol address:

```
(ADDRESS=(PROTOCOL=tcp)(HOST=host3)(PORT=1521))
```

Example 6-5 Client Load Balancing in tnsnames.ora

This example illustrates client load balancing among two Oracle Connection Managers and two protocol addresses:

```
sample2=
 (DESCRIPTION=
   (LOAD_BALANCE=on)                                    # 1
   (FAILOVER=on)
   (ADDRESS_LIST=
     (SOURCE_ROUTE=yes)
     (ADDRESS=(PROTOCOL=tcp)(HOST=host1)(PORT=1630))    # 2
     (ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1521)))
  (ADDRESS_LIST=
     (SOURCE_ROUTE=yes)
     (ADDRESS=(PROTOCOL=tcp)(HOST=host3)(port=1630))
     (ADDRESS=(PROTOCOL=tcp)(HOST=host4)(port=1521)))
   (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))  # 3
```

The previous syntax does the following:

1. The client selects an `ADDRESS_LIST` at random and fails over to the other address if the chosen `ADDRESS_LIST` fails. This is indicated if you set the `LOAD_BALANCE` and `FAILOVER` parameters to `on`.
2. When an `ADDRESS_LIST` is chosen, the client first connects to Oracle Connection Manager using the Oracle Connection Manager protocol address that uses port 1630 as is indicated for the `ADDRESS_LIST`.
3. Oracle Connection Manager then connects to the database service using the protocol address that is indicated for the `ADDRESS_LIST`.

6.6 Connect Descriptor Descriptions

Specify connect descriptors using the `DESCRIPTION` parameter. Identify multiple connect descriptors with the `DESCRIPTION_LIST` parameter.

- DESCRIPTION_LIST
  The `DESCRIPTION_LIST` parameter of the `tnsnames.ora` file defines a list of connect descriptors for a particular net service name.

- DESCRIPTION
  Use the `tnsnames.ora` file `DESCRIPTION` parameter to specify connect descriptor containers.

6.6.1 DESCRIPTION_LIST

The `DESCRIPTION_LIST` parameter of the `tnsnames.ora` file defines a list of connect descriptors for a particular net service name.

Purpose

To define a list of connect descriptors for a particular net service name.

Example 6-6 Example

```
net_service_name=
(DESCRIPTION_LIST=
 (DESCRIPTION=
  (ADDRESS=...)
  (CONNECT_DATA=(SERVICE_NAME=sales.example.com)))
 (DESCRIPTION=
```

6.6.2 DESCRIPTION

Use the `tnsnames.ora` file `DESCRIPTION` parameter to specify connect descriptor containers.

Purpose

To specify a container for a connect descriptor.

Usage Notes

When using more than one `DESCRIPTION` parameter, place the parameters under the `DESCRIPTION_LIST` parameter.

Example 6-7 DESCRIPTION Parameter Example

```
net_service_name=
(DESCRIPTION=
  (ADDRESS=...)
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

6.7 Protocol Addresses

Learn about Oracle Net Services protocol address parameters.

The protocol address section of a `tnsnames.ora` file specifies listener protocol addresses. If there is only one listener protocol address, then use the `ADDRESS` parameter. If there is more than one address, then use the `ADDRESS_LIST` parameter.

- ADDRESS
  The `tnsnames.ora` parameter `ADDRESS` specifies protocol addresses with the `ADDRESS_LIST` for multiple addresses or with the `DESCRIPTION` parameter for one listener.
- HTTPS_PROXY
  Learn to use the `tnsnames.ora` parameter `HTTPS_PROXY` to specify HTTP proxy host names to tunnel Transport Layer Security (TLS) client connections.
- HTTPS_PROXY_PORT
  Learn how to use the `tnsnames.ora` parameter `HTTPS_PROXY_PORT` to specify forward HTTP proxy host ports for tunneling Transport Layer Security (TLS) client connections.
- ADDRESS_LIST
  The `ADDRESS_LIST` networking parameter specifies the number of protocol addresses.

6.7.1 ADDRESS

The `tnsnames.ora` parameter `ADDRESS` specifies protocol addresses with the `ADDRESS_LIST` for multiple addresses or with the `DESCRIPTION` parameter for one listener.

Purpose

To specify one listener protocol address.

Usage Notes

Put this parameter under either the `ADDRESS_LIST` parameter or the `DESCRIPTION` parameter.

ADDRESS Parameter Example

```
net_service_name=
(DESCRIPTION=
 (ADDRESS=(PROTOCOL=tcp)(HOST=sales-svr)(PORT=1521))
 (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
```

6.7.2 HTTPS_PROXY

Learn to use the `tnsnames.ora` parameter `HTTPS_PROXY` to specify HTTP proxy host names to tunnel Transport Layer Security (TLS) client connections.

Purpose

To specify HTTP proxy host names for tunneling your TLS client connections.

Usage Notes

Your clients can tunnel secure connections over HTTP proxy using the `HTTP CONNECT` method. This helps access the public cloud database service because it eliminates the need to open an outbound port on a client-side firewall. This parameter is applicable only to the connect descriptors where `PROTOCOL=TCPS`. This is similar to the web browser setting for intranet users who want to connect to internet hosts. You can increase the forward web proxy read timeout for requests to a higher value depending on client queries. Otherwise, the forward web proxy closes the connection assuming that no requests are made from the client.

A successful connection depends on your specific proxy configurations. The performance of your data transfers depend on the proxy capacity. Oracle recommends against using this feature in production environments where performance is critical.

Configuring `tnsnames.ora` for an HTTP proxy may not be secure enough, depending your organization's network configuration and security policies. For example, some networks require a user name and password for the HTTP proxy.

Oracle Client versions earlier than 18c does not support connections through HTTP proxy.

Contact your network administrator to open outbound connections to hosts that are in the `oraclecloud.com` domain by using the relevant port, without going through an HTTP proxy. For example, port 1522.

Default

None

Values

An HTTP proxy host name that can make an outbound connection to internet hosts.

Example

```
HTTPS_PROXY=www-proxy.example.com
```

6.7.3 HTTPS_PROXY_PORT

Learn how to use the `tnsnames.ora` parameter `HTTPS_PROXY_PORT` to specify forward HTTP proxy host ports for tunneling Transport Layer Security (TLS) client connections.

Purpose

To specify forward HTTP proxy host port for tunneling TLS client connections.

Usage Notes

It forwards the HTTP proxy host port that receives the HTTP CONNECT method. Use this parameter with `HTTPS_PROXY_PORT`. The value for the `HTTPS_PROXY_PORT` parameter takes effect only when you set `SQLNET.USE_HTTPS_PROXY=1` set in your `sqlnet.ora` file.

Default

none

Values

port number

Example

```
HTTPS_PROXY_PORT=80
```

6.7.4 ADDRESS_LIST

The ADDRESS_LIST networking parameter specifies the number of protocol addresses.

Purpose

To define a list of protocol addresses.

Usage Notes

If there is only one listener protocol address, then ADDRESS_LIST is not necessary.

Put this parameter either under the DESCRIPTION parameter or the DESCRIPTION_LIST parameter.

Example

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

6.8 Optional Parameters for Address Lists

For multiple addresses, you can use the optional parameters to configure address lists.

- ENABLE
- EXPIRE_TIME
  Use the EXPIRE_TIME parameter to specify how often, in minutes, to verify that the remote server connection is active.
- FAILOVER
- LOAD_BALANCE
- RECV_BUF_SIZE
  Use the RECV_BUF_SIZE parameter to specify buffer space for session receive operations.
- SDU
- SEND_BUF_SIZE
  Use the SEND_BUF_SIZE parameter to specify buffer space for session send operations.
- SOURCE_ROUTE
- TYPE_OF_SERVICE

6.8.1 ENABLE

Purpose

To allow the caller to detect a terminated remote server, typically it takes 2 hours or more to notice.

Usage Notes

The keepalive feature on the supported TCP transports can be enabled for a net service client by putting `(ENABLE=broken)` under the `DESCRIPTION` parameter in the connect string. On the client side, the default for `tcp_keepalive` is `off`. Operating system TCP configurables, which vary by platform, define the actual keepalive timing details.

Values

`broken`

Example

```
net_service_name=
 (DESCRIPTION=
  (ENABLE=broken)
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
```

Although the preceding example has multiple addresses, the `ADDRESS_LIST` parameter was not used. This is because the `ADDRESS_LIST` parameter is not mandatory.

6.8.2 EXPIRE_TIME

Use the `EXPIRE_TIME` parameter to specify how often, in minutes, to verify that the remote server connection is active.

Purpose

To specify time intervals, in minutes, for how often to verify that the remote server connection is active.

Usage Notes

Oracle Net Services tunes the TCP keepalive parameters so that probes are sent after an idle activity.

Limitations on using the terminated connection detection feature are:

- You cannot use it on bequeathed connections.
- Though very small, a probe packet generates additional traffic that may degrade your network performance.
- Depending on your operating system, the server may need to perform additional processing to distinguish the connection probing event from other events. This can also result in a degraded network performance.

Default

0

Minimum Value

0

Recommended Value

10

Example

```
net_service_name=
   (DESCRIPTION=
     (EXPIRE_TIME=10)
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521)))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
```

6.8.3 FAILOVER

Purpose

To enable or disable connect-time failover for multiple protocol addresses.

Usage Notes

When you set the parameter to `on`, `yes`, or `true`, Oracle Net fails over at connect time to a different address if the first protocol address fails. When you set the parameter to `off`, `no`, or `false`, Oracle Net tries one protocol address.

Put this parameter under the `DESCRIPTION_LIST` parameter, the `DESCRIPTION` parameter, or the `ADDRESS_LIST` parameter.

> **Note:**
>
> Do not set the `GLOBAL_DBNAME` parameter in the `SID_LIST_`*`listener_name`* section of the `listener.ora`. A statically configured global database name disables connect-time failover.

Default

`on` for the `DESCRIPTION_LIST`, `DESCRIPTION`, and `ADDRESS_LIST` parameters

Values

- `yes` | `on` | `true`
- `no` | `off` | `false`

Example

```
net_service_name=
 (DESCRIPTION=
  (FAILOVER=on)
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

6.8.4 LOAD_BALANCE

Purpose

To enable or disable client load balancing for multiple protocol addresses.

Usage Notes

When you set the parameter to `on`, `yes`, or `true`, Oracle Net goes through the list of addresses in a random sequence, balancing the load on the various listener or Oracle Connection Manager protocol addresses. When you set the parameter to `off`, `no`, or `false`, Oracle Net tries the protocol addresses sequentially until one succeeds.

Put this parameter under the `DESCRIPTION_LIST` parameter, the `DESCRIPTION` parameter, or the `ADDRESS_LIST` parameter.

Default

`on` for `DESCRIPTION_LIST`

Values

- `yes` | `on` | `true`
- `no` | `off` | `false`

Example

```
net_service_name=
 (DESCRIPTION=
  (LOAD_BALANCE=on)
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
```

6.8.5 RECV_BUF_SIZE

Use the `RECV_BUF_SIZE` parameter to specify buffer space for session receive operations.

Purpose

To specify, in bytes, the buffer space for receive operations of sessions.

Usage Notes

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

Put this parameter under the `DESCRIPTION` parameter or at the end of the protocol address.

Setting this parameter in the connect descriptor for a client overrides the RECV_BUF_SIZE parameter at the client-side `sqlnet.ora` file.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to the operating system-specific documentation for additional information about additional protocols.

Default

The default value for this parameter is specific to the operating system. The default for the Linux 2.6 operating system is 87380 bytes.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521)
        (RECV_BUF_SIZE=11784))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)
        (RECV_BUF_SIZE=11784))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)))
net_service_name=
 (DESCRIPTION=
   (RECV_BUF_SIZE=11784)
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=hr1-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=tcp)(HOST=hr2-server)(PORT=1521)))
   (CONNECT_DATA=
     (SERVICE_NAME=hr.us.example.com)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.8.6 SDU

Purpose

To instruct Oracle Net to optimize the transfer rate of data packets being sent across the network with a specified session data unit (SDU) size.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter.

Setting this parameter in the connect descriptor for a client overrides the `DEFAULT_SDU_SIZE` parameter at client-side `sqlnet.ora` file.

Default

8192 bytes (8 KB)

Values

512 to 2097152 bytes.

Example

```
net_service_name=
 (DESCRIPTION=
  (SDU=8192)
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com))
```

6.8.7 SEND_BUF_SIZE

Use the `SEND_BUF_SIZE` parameter to specify buffer space for session send operations.

Purpose

To specify, in bytes, the buffer space for send operations of sessions.

Usage Notes

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

Put this parameter under the `DESCRIPTION` parameter or at the end of the protocol address.

Setting this parameter in the connect descriptor for a client overrides the SEND_BUF_SIZE parameter at the client-side `sqlnet.ora` file.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to the operating system-specific documentation for information about additional protocols.

Default

The default value for this parameter is operating system specific. The default for the Linux 2.6 operating system is 16 KB.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521)
        (SEND_BUF_SIZE=11784))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)
        (SEND_BUF_SIZE=11784)))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)))
net_service_name=
 (DESCRIPTION=
   (SEND_BUF_SIZE=11784)
   (ADDRESS_LIST=
```

```
    (ADDRESS=(PROTOCOL=tcp)(HOST=hr1-server)(PORT=1521)
    (ADDRESS=(PROTOCOL=tcp)(HOST=hr2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=hr.us.example.com)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.8.8 SOURCE_ROUTE

Purpose

To enable routing through multiple protocol addresses.

Usage Notes

When you set this parameter to `on` or `yes`, Oracle Net uses each address in order until the destination is reached.

To use Oracle Connection Manager, an initial connection from the client to Oracle Connection Manager is required, and a second connection from Oracle Connection Manager to the listener is required.

Put this parameter under either the `DESCRIPTION_LIST` parameter, the `DESCRIPTION` parameter, or the `ADDRESS_LIST` parameter.

Default

off

Values

- `yes` | `on`
- `no` | `off`

Example

```
net_service_name=
 (DESCRIPTION=
  (SOURCE_ROUTE=on)
  (ADDRESS=(PROTOCOL=tcp)(HOST=cman-pc)(PORT=1630))
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
```

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for complete configuration information

6.8.9 TYPE_OF_SERVICE

Purpose

To specify the type of service to use for an Oracle Rdb database.

Usage Notes

This parameter should only be used if the application supports both an Oracle Rdb and Oracle database service, and you want the application to load balance between the two.

Put this parameter under the `DESCRIPTION` parameter.

Example

```
net_service_name=
 (DESCRIPTION_LIST=
  (DESCRIPTION=
   (ADDRESS=...)
   (CONNECT_DATA=
    (SERVICE_NAME=generic)
    (RDB_DATABASE=[.mf]mf_personal.rdb)
    (GLOBAL_NAME=alpha5))
   (TYPE_OF_SERVICE=rdb_database))
  (DESCRIPTION=
   (ADDRESS=...)
   (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com))
   (TYPE_OF_SERVICE=oracle11_database)))
```

6.9 Connection Data Section

Learn how to configure network connections with protocol addresses.

A network object is identified by a protocol address. When a connection is made, the client and the receiver of the request (listener or Oracle Connection Manager) are configured with identical protocol addresses. The client uses this address to send the connection request to a particular network object location, and the recipient "listens" for requests on this address, and grants a connection based on its address information matching the client information.

- CONNECT_DATA
  Use the `CONNECT_DATA` parameter to define the connection service.
- COLOCATION_TAG
- CONNECTION_ID_PREFIX
  Use this parameter to add application specific ID to connection identifier.
- FAILOVER_MODE
- GLOBAL_NAME
- HS
- INSTANCE_NAME

- POOL_BOUNDARY
  Use the POOL_BOUNDARY parameter to enable implicit connection pooling with either Database Resident Connection Pooling (DRCP) or Proxy Resident Connection Pooling (PRCP).
- POOL_CONNECTION_CLASS
  Use this parameter to explicitly name the connection class for Database Resident Connection Pooling (DRCP) connections.
- POOL_NAME
  Use the POOL_NAME parameter to specify a pool name for multi-pool Database Resident Connection Pooling (DRCP) connections.
- POOL_PURITY
  Use this parameter to specify if an application needs a new session that is not tainted with any prior session state or to reuse a previous session.
- RDB_DATABASE
- SHARDING_KEY
  Use the SHARDING_KEY parameter to route the database connection request to an appropriate shard.
- SHARDING_KEY_ID
  Use this parameter to route the database connection request to a shard using the unique SHA-256 ID of a sharding key (instead of a sharding key value).
- SUPER_SHARDING_KEY
  Use the SUPER_SHARDING_KEY parameter in the case of composite sharding to route the database request to a collection of shards (shardspace).
- SERVER
- SERVICE_NAME
- TUNNEL_SERVICE_NAME
  Set this parameter to identify the client CMAN.

6.9.1 CONNECT_DATA

Use the CONNECT_DATA parameter to define the connection service.

Purpose

To define the service to which you want to connect, such as SERVICE_NAME.

Usage Notes

Put this parameter under the DESCRIPTION parameter. CONNECT_DATA permits additional parameters as listed in Connection Data Section.

Example

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

6.9.2 COLOCATION_TAG

Purpose

To direct the listener to route all connections with the same `colocation_tag` to the same database instance.

Usage Notes

Use this parameter with the `CONNECT_DATA` parameter.

The parameter value must be an alphanumeric string.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
    (ADDRESS=...)
    (ADDRESS=...))
 (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    (COLOCATION_TAG=abc)))
```

> **Note:**
>
> Under certain conditions, such as, when maximum load of an instance is reached or when new instances are added or deleted for a service, the colocation of client connections that have the same `colocation_tag` to the same database instance may not be consistent.

6.9.3 CONNECTION_ID_PREFIX

Use this parameter to add application specific ID to connection identifier.

Usage Notes

Put this parameter under the `CONNECT_DATA` parameter.

Example

```
net_service_name=
(DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
```

```
(SERVICE_NAME=sales.us.example.com)
((CONNECTION_ID_PREFIX=value)))
```

> **Note:**
>
> The CONNECTION_ID_PREFIX value is appended internally to a system generated connection ID value and sent as CONNECTION_ID in connect string. The CONNECTION_ID_PREFIX must be an 8-byte alphanumeric identifier limited to the following [a...z] [A...Z] [0...9] _ character set.

Related Topics

- Permitted Listener and Net Service Name Character Set
  Create listener names and net service names for clients that comply with Oracle Net Services character set requirements.

6.9.4 FAILOVER_MODE

Purpose

To instruct Oracle Net to fail over to a different listener if the first listener fails during run time.

Usage Notes

Depending upon the configuration, the session or any SELECT statements which were in progress are automatically failed over.

This type of failover is called Transparent Application Failover (TAF) and should not be confused with the connect-time failover FAILOVER parameter.

Put this parameter under the CONNECT_DATA parameter.

Additional Parameters

FAILOVER_MODE supports the following parameters:

- BACKUP: Specifies the failover node by its net service name. A separate net service name must be created for the failover node.
- TYPE: Specifies the type of failover. Three types of Oracle Net failover functionality are available by default to Oracle Call Interface (OCI) applications:
  - SESSION: Fails over the session. For example, if a user's connection is lost, then a new session is automatically created for the user on the backup. This type of failover does not attempt to recover selects.
  - SELECT: Allows users with open cursors to continue fetching them after failure. However, this mode involves overhead on the client side in normal select operations.
  - NONE: This is the default, in which no failover functionality is used. This can also be explicitly specified to prevent failover from happening.
- METHOD: Specifies how fast failover is to occur from the primary node to the backup node:
  - BASIC: Establishes connections at failover time. This option requires almost no work on the backup database server until failover time.

  - PRECONNECT: Pre-establishes connections. This provides faster failover but requires that the backup instance be able to support all connections from every supported instance.

- TRANSACTION: Allows the database to complete the current database transaction following a recoverable error. This parameter is used with the COMMIT_OUTCOME=TRUE parameter.
- RETRIES: Specifies the number of times to attempt to connect after a failover. If DELAY is specified, then RETRIES defaults to five retry attempts.
- DELAY: Specifies the amount of time in seconds to wait between connect attempts. If RETRIES is specified, then DELAY defaults to one second.

> **Note:**
>
> If a callback function is registered, then RETRIES and DELAY parameters are ignored.

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.9.5 GLOBAL_NAME

Purpose

To identify the Oracle Rdb database.

Usage Notes

Put this parameter under the CONNECT_DATA parameter.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
    (ADDRESS=...)
    (ADDRESS=...))
   (CONNECT_DATA=
    (SERVICE_NAME=generic)
    (RDB_DATABASE=[.mf]mf_personal.rdb)
    (GLOBAL_NAME=alpha5)))
```

6.9.6 HS

Purpose

To direct Oracle Net to connect to a non-Oracle system through Heterogeneous Services.

Usage Notes

Put this parameter under the CONNECT_DATA parameter.

Default

None

Values

`ok`

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
    (SID=sales6)
    )
(HS=ok))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.9.7 INSTANCE_NAME

Purpose

To identify the database instance to access.

Usage Notes

Set the value to the value specified by the `INSTANCE_NAME` parameter in the initialization parameter file.

Put this parameter under the `CONNECT_DATA` parameter.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_NAME=sales1)))
```

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for additional information about the use of `INSTANCE_NAME`

6.9.8 POOL_BOUNDARY

Use the `POOL_BOUNDARY` parameter to enable implicit connection pooling with either Database Resident Connection Pooling (DRCP) or Proxy Resident Connection Pooling (PRCP).

Purpose

To enable implicit connection pooling with either DRCP (on the database server side) or PRCP (for Oracle Connection Manager in Traffic Director Mode). This setting specifies a time boundary to release an application session back to the DRCP or PRCP pool. You can set implicit connection pooling at a statement or transaction boundary.

If you omit this parameter setting, then implicit connection pooling is disabled.

Usage Notes

- Specify this parameter under the `CONNECT_DATA` section of the connect string in the `tnsnames.ora` file, Easy Connect syntax, or directly as part of the command-line connect string.
- This parameter is applicable to pooled connections only. In the application tier, to configure DRCP or PRCP, the client must specify the server type as `POOLED` using the `SERVER=POOLED` setting.
- When set to `STATEMENT`, a session is released back to the connection pool when the session is implicitly stateless. A session is implicitly stateless when all open cursors in a session have been fetched through to completion and there are no active transactions, temporary tables, or temporary LOBs.

  Use the `STATEMENT` value for applications that create minimal session states from partially fetched cursors, temporary LOBs, and global or private temporary tables.
- When set to `TRANSACTION`, a session is released back to the connection pool when a transaction ends implicitly or explicitly, or when a transaction is not available and the session is stateless. The release to the pool closes any active cursors, temporary tables, and temporary LOBs.

  Use the `TRANSACTION` value for applications that create many session states from partially fetched cursors, temporary LOBs, and global or private temporary tables and for applications that have occasional commits or rollbacks of transactions.
- If an application uses session pooling APIs along with the `POOL_BOUNDARY=STATEMENT` or `POOL_BOUNDARY=TRANSACTION` attribute in the connect string, then the connect string setting takes precedence over the pooling APIs. The session is released back to the connection pool using the statement or transaction boundary directive, overriding the session release API call.

Default

None

Values and Examples

Value	Example

`STATEMENT`: To set implicit connection pooling at a statement boundary	Easy connect string:
	
	`sales1-server:1521/sales.us.example.com:pooled?pool_boundary=statement`
	
	Connect descriptor in `tnsnames.ora`:
	
	`inst1=`
	`  (DESCRIPTION=`
	`  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))`
	`  (CONNECT_DATA=`
	`  (SERVICE_NAME=sales.us.example.com)`
	`  (SERVER=POOLED)`
	`  (POOL_BOUNDARY=STATEMENT))`
	`  )`

`TRANSACTION`: To set implicit connection pooling at a transaction boundary	Easy connect string:
	
	`sales1-server:1521/sales.us.example.com:pooled?pool_boundary=transaction`
	
	Connect descriptor in `tnsnames.ora`:
	
	`inst1=`
	`  (DESCRIPTION=`
	`  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))`
	`  (CONNECT_DATA=`
	`  (SERVICE_NAME=sales.us.example.com)`
	`  (SERVER=POOLED)`
	`  (POOL_BOUNDARY=TRANSACTION))`
	`  )`

The following example of a connect descriptor in `tnsnames.ora` shows how to set implicit connection pooling in different modes:

```
inst1s=
    (DESCRIPTION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
        (CONNECT_DATA=
            (SERVICE_NAME=sales.us.example.com)
            (SERVER=POOLED)
            (POOL_BOUNDARY=STATEMENT))
    )
inst1t=
```

```
(DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=proxy-server)(PORT=1522))
    (CONNECT_DATA=
        (SERVICE_NAME=sales.us.example.com)
        (SERVER=POOLED)
        (POOL_BOUNDARY=TRANSACTION))
)
```

For the `inst1s` application instance, `POOL_BOUNDARY=STATEMENT` specifies that implicit connection pooling applies at a statement boundary. `HOST=sales-server` and `PORT=1521` specify a DRCP connection through the database server.

For the `inst1t` application instance, `POOL_BOUNDARY=TRANSACTION` specifies that implicit connection pooling applies at a transaction boundary. `HOST=proxy-server` and `PORT=1522` specify a PRCP connection through Oracle Connection Manager in Traffic Director Mode.

Related Topics

- *Oracle Database Development Guide*
- *Oracle Database Net Services Administrator's Guide*

6.9.9 POOL_CONNECTION_CLASS

Use this parameter to explicitly name the connection class for Database Resident Connection Pooling (DRCP) connections.

Usage Notes

Add this parameter in the connect string under `CONNECT_DATA` section of the connect identifier. This parameter takes precedence and overrides the properties programmatically set by the application using this connect string.

Example

```
ServerPool =
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp) (HOST=sales-svr) (PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(SERVER=POOLED)
(POOL_CONNECTION_CLASS=value)))
```

Related Topics

- POOL_PURITY
  Use this parameter to specify if an application needs a new session that is not tainted with any prior session state or to reuse a previous session.

6.9.10 POOL_NAME

Use the `POOL_NAME` parameter to specify a pool name for multi-pool Database Resident Connection Pooling (DRCP) connections.

Purpose

To specify a pool name for each connection in multi-pool DRCP. This enables client applications or services to establish connections from specific DRCP pools.

With this setting, DRCP marks the connection against the appropriate pooled server. If you do not set this parameter, then the connection is established from the default pool if it is active.

Usage Notes

Use this parameter under the `CONNECT_DATA` section of the `tnsnames.ora` file or directly as part of the command-line connect string.

When the client connects with `POOL_NAME=`*`pool_name`* along with the `SERVER=POOLED` setting, DRCP checks whether a DRCP pool with the given pool name is added in the database:

- If the pool does not exist, then an error is returned.
- If the pool with the given pool name exists and is active, then the connection is established from the specified pooled server.
- If the pool with the given pool name exists but is inactive, then an error is returned for any new connections attempted. If connections are already established and the pool is made inactive, then those connections are handled based on the `drainTime` parameter used in the `stop_pool()` procedure (called to stop the DRCP pool).

Database administrators add a DRCP pool at the PDB or CDB level. Static data dictionary view `DBA_CPOOL_INFO` contains information about existing DRCP pools. For example, this query lists all the available active pools:

```
SELECT * FROM DBA_CPOOL_INFO WHERE status='ACTIVE';
```

Value

DRCP pool name

Default

None

Example

```
(DESCRIPTION=
  (ADDRESS=
    (PROTOCOL=TCP)
    (HOST=sales-svr)
    (PORT=1522)
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
```

```
    (SERVER=POOLED)
    (POOL_NAME=mypool)
  )
)
```

Related Topics

- *Oracle Database Development Guide*

6.9.11 POOL_PURITY

Use this parameter to specify if an application needs a new session that is not tainted with any prior session state or to reuse a previous session.

Usage Notes

Starting with Oracle Database 21c, you can configure Database Resident Connection Pooling (DRCP) for specific pluggable databases (PDBs). Add this parameter in the connect string under `CONNECT_DATA` section of the connect identifier to set purity attributes to a DRCP connection request.

This parameter takes precedence and overrides the properties programmatically set by the application using this connect string.

Values

`NEW/SELF`

Example

```
ServerPool =
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp) (HOST=sales-svr) (PORT=1521))
   (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(SERVER=POOLED)
(POOL_PURITY=value)))
```

Related Topics

- POOL_CONNECTION_CLASS
  Use this parameter to explicitly name the connection class for Database Resident Connection Pooling (DRCP) connections.

6.9.12 RDB_DATABASE

Purpose

To specify the file name of an Oracle Rdb database.

Usage Notes

Put this parameter under the `CONNECT_DATA` parameter.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (RDB_DATABASE= [.mf]mf_personal.rdb)))
```

6.9.13 SHARDING_KEY

Use the `SHARDING_KEY` parameter to route the database connection request to an appropriate shard.

Purpose

To specify the value of a sharding key. Based on the value specified during a database connection request, the request is directly routed to the appropriate shard.

Usage Notes

You specify this parameter under the `CONNECT_DATA` section of a connect string or `tnsnames.ora` file.

Use the `SHARDING_KEY` parameter to specify a sharding key in simplified text format. This parameter supports only ASCII character set and not special characters. The following data types are supported for a sharding key:

- `NUMBER`
- `INTEGER`
- `SMALLINT`
- `RAW`
- `NVARCHAR`
- `NVARCHAR2`
- `NCHAR`
- `DATE`
- `TIMESTAMP`

Use the `SHARDING_KEY_B64` parameter to specify the base64-encoded binary representation of a sharding key. This parameter supports these special characters: `"` quotation mark `,` comma `( )` close parenthesis `+` plus sign)

Values

The fields for base64-encoded values (`*_B64`) start with a header, which is a sequence of space-separated integer values:

```
(CONNECT_DATA=
  (SHARDING_KEY_B64=
```

```
    [version][type][key column 1 type identifier][key column 2 type
identifier] ... ,[base64 string],[base64 string],[base64
string],...))...
```

In the above syntax:

- Parts of the compound key are separated with a comma.
- *version* specifies the version number of base64 representation. Currently, only version 1 is supported, and thus the supported *version* value is 1.
- type specifies the character set string and its encoding information. The supported type values are:

Value	Character Set String	Encoding Scheme
0	String contains hash value.	Character values are encoded in AL32UTF8 (for VARCHAR) and AL16UTF16 (for NVARCHAR).
1	String does not contain hash value.	
2	String does not contain hash value.	Character values are encoded in database encoding, which may be specific for each column.
3	String contains hash value.	
4	String contains only hash value.	

- *key column type identifier* specifies the data types. The supported *key column type identifier* values are:

Value	Data Type
1	VARCHAR, NVARCHAR, CHAR, NCHAR
2	NUMBER
6	NUMBER with length in first byte
12	DATE
23	RAW
180	TIMESTAMP

- The header is terminated by a comma and is followed by *base64 string*. *base64 string* is a comma-separated list of the base64-encoded value string. The hash value, if available, is the last value in the list.

Example 6-8

In the following sample connect string, the SHARDING_KEY parameter value is specified in simplified text format:

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(host=sales-east1)(port=1522))
    (ADDRESS=(host=sales-east2)(port=1522))
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
```

```
    (SHARDING_KEY=40598230)
  )
 )
```

Example 6-9

In the following sample connect string, the `SHARDING_KEY_B64` parameter value is encoded to base64 binary representation:

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(host=sales-east1)(port=1522))
    (ADDRESS=(host=sales-east2)(port=1522))
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    (SHARDING_KEY_B64=1 1 2,VVM=,OTQwMDI=)
  )
 )
```

Related Topics

- SUPER_SHARDING_KEY
  Use the `SUPER_SHARDING_KEY` parameter in the case of composite sharding to route the database request to a collection of shards (shardspace).
- SHARDING_KEY_ID
  Use this parameter to route the database connection request to a shard using the unique SHA-256 ID of a sharding key (instead of a sharding key value).
- *Oracle Database Net Services Administrator's Guide*

6.9.14 SHARDING_KEY_ID

Use this parameter to route the database connection request to a shard using the unique SHA-256 ID of a sharding key (instead of a sharding key value).

Purpose

If you are using hashing to encrypt sharding keys, then use this parameter to specify the unique SHA256 hash value assigned to a sharding key. Based on the value specified during the database connection request, the request is directly routed to the appropriate shard.

Usage Notes

You use this parameter for directory-based sharding.

Set this parameter under the `CONNECT_DATA` section of a connect string or `tnsnames.ora` file.

Value

SHA256 hash value of a sharding key in simplified text format. This value supports only the `RAW` data type.

You must enclose the value in single quotation marks.

Default

None

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
    (ADDRESS=(host=sales-east1)(port=1522))
    (ADDRESS=(host=sales-east2)(port=1522))
   )
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)

(SHARDING_KEY_ID='7E01C6D3F5AF3116668AFB6B2376DAA457165A34020617884C216
F1ADAA25C7B')
   )
 )
```

Related Topics

- SHARDING_KEY
  Use the `SHARDING_KEY` parameter to route the database connection request to an appropriate shard.
- SUPER_SHARDING_KEY
  Use the `SUPER_SHARDING_KEY` parameter in the case of composite sharding to route the database request to a collection of shards (shardspace).
- *Oracle Database Net Services Administrator's Guide*

6.9.15 SUPER_SHARDING_KEY

Use the `SUPER_SHARDING_KEY` parameter in the case of composite sharding to route the database request to a collection of shards (shardspace).

Purpose

To specify a shardspace key for a collection of shards. A shardspace is set of shards that store data that corresponds to a range or list of key values. Based on the value specified during a database connection request, the request is directly routed to an appropriate shardspace.

Usage Notes

You specify this parameter under the `CONNECT_DATA` section of a connect string or `tnsnames.ora` file.

Use the `SUPER_SHARDING_KEY` parameter to specify a shardspace key for a collection of shards in simplified text format. This parameter supports only ASCII character set and not special characters. The supported data types for a super sharding key are the same as those for a sharding key.

Use the SUPER_SHARDING_KEY_B64 parameter to specify the base64-encoded binary representation of a shardspace key. This parameter supports special characters (such as " quotation mark , comma ( ) close parenthesis + plus sign).

Values

The fields for base64-encoded values (*_B64) start with a header, which is a sequence of space-separated integer values:

```
(CONNECT_DATA=(SUPER_SHARDING_KEY_B64=[version] [type] [integer literal]
[integer literal] ... ,[base64 binary],[base64 binary],[base64
binary],...))...
```

For details on each of the base64-encoded header fields, see SHARDING_KEY.

Example 6-10

In the following sample connect string, the SHARDING_KEY and SUPER_SHARDING_KEY parameter values are specified in simplified text format:

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(host=sales-east1)(port=1522))
    (ADDRESS=(host=sales-east2)(port=1522))
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    ((SHARDING_KEY=40598230)(SUPER_SHARDING_KEY=gold))
  )
 )
```

Example 6-11

In the following sample connect string, the SHARDING_KEY_B64 and SUPER_SHARDING_KEY_B64 parameter values are encoded to base64 binary representation:

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(host=sales-east1)(port=1522))
    (ADDRESS=(host=sales-east2)(port=1522))
  )
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    ((SHARDING_KEY_B64=1 1 2,VVM=,OTQwMDI=)(SUPER_SHARDING_KEY_B64=1
1,BBWEPGRBBDOEMGQW))
  )
 )
```

Related Topics

- SHARDING_KEY
  Use the SHARDING_KEY parameter to route the database connection request to an appropriate shard.

- SHARDING_KEY_ID
  Use this parameter to route the database connection request to a shard using the unique SHA-256 ID of a sharding key (instead of a sharding key value).
- *Oracle Database Net Services Administrator's Guide*

6.9.16 SERVER

Purpose

To direct the listener to connect the client to a specific type of service handler.

Usage Notes

Put this parameter under the `CONNECT_DATA` parameter.

Values

- `dedicated` to specify whether client requests be served by dedicated server.
- `shared` to specify whether client requests be served by a dispatcher or shared server.
- `pooled` to get a connection from the connection pool if database resident connection pooling is enabled on the server.

> **Note:**
>
> - Shared server must be configured in the database initialization file in order for the client to connect to the database with a shared server process.
> - The USE_DEDICATED_SERVER parameter in the `sqlnet.ora` file overrides this parameter.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    (SERVER=dedicated)))
```

6.9.17 SERVICE_NAME

Purpose

To identify the Oracle Database database service to access.

Usage Notes

Set the value to a value specified by the `SERVICE_NAMES` parameter in the initialization parameter file.

Put this parameter under the `CONNECT_DATA` parameter.

Example

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.9.18 TUNNEL_SERVICE_NAME

Set this parameter to identify the client CMAN.

Purpose

The server CMAN listener will route the connection to a gateway that has a tunnel connection to the requested client ID.

Usage Notes

Put this parameter under the `CONNECT_DATA` parameter.

Example

```
net_service_name=
(DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=...)
     (ADDRESS=...))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (TUNNEL_SERVICE_NAME=south)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.10 Security Section

The security section of the `tnsnames.ora` file specifies these security-related parameters for use with Oracle security features.

- AUTHENTICATION_SERVICE
  Use the `tnsnames.ora` parameter `AUTHENTICATION_SERVICE` to enable one or more authentication services.
- AZURE_DB_APP_ID_URI
  Use the `AZURE_DB_APP_ID_URI` parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).
- CLIENT_CERTIFICATE
  Use the `CLIENT_CERTIFICATE` parameter to specify the file system path to a client certificate that authenticates the database client.
- CLIENT_ID
  Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.
- IGNORE_ANO_ENCRYPTION_FOR_TCPS
  The `IGNORE_ANO_ENCRYPTION_FOR_TCPS` parameter specifies if the `SQLNET.ENCRYPTION_CLIENT` parameter should be ignored for this specific TNS alias.
- KERBEROS5_CC_NAME
  Use the `tnsnames.ora` parameter `KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- KERBEROS5_PRINCIPAL
  Use the `KERBEROS5_PRINCIPAL` parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.
- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.
- OCI_CONFIG_FILE
  Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.
- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.
- OCI_IAM_URL
  Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.
- OCI_TENANCY
  Use the `OCI_TENANCY` parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.
- PASSWORD_AUTH
- REDIRECT_URI
  Use the `REDIRECT_URI` parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.

- SECURITY
  Use the `SECURITY` parameter to change the security properties of a connection.
- SSL_CERTIFICATE_ALIAS
  Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_ALIAS` to specify the alias of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.
- SSL_CERTIFICATE_THUMBPRINT
  Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_THUMBPRINT` to specify the thumbprint of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.
- SSL_CLIENT_AUTHENTICATION
  Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).
- SSL_SERVER_CERT_DN
  Use the `SSL_SERVER_CERT_DN` parameter to specify the distinguished name (DN) of the database server.
- SSL_SERVER_DN_MATCH
  Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.
- SSL_VERSION
  Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.
- TENANT_ID
  Use the `TENANT_ID` parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.
- TOKEN_AUTH
- TOKEN_LOCATION
  Use the `TOKEN_LOCATION` parameter to specify the directory location where token file is stored for token-based authentication.
- WALLET_LOCATION
  Use the `WALLET_LOCATION` parameter in the `tnsnames.ora` file to specify different locations where Oracle wallets are stored.

6.10.1 AUTHENTICATION_SERVICE

Use the `tnsnames.ora` parameter `AUTHENTICATION_SERVICE` to enable one or more authentication services.

Purpose

To enable one or more authentication services. If you have installed authentication, then Oracle recommends that you set `AUTHENTICATION_SERVICE` to either `NONE` or to one of the listed authentication methods.

Usage Notes

Use this parameter in the `SECURITY` section of the `tnsnames.ora` file or directly as part of the connect string. You can also set it in the `sqlnet.ora` file. Note that this parameter is called

SQLNET.AUTHENTICATION_SERVICES in sqlnet.ora. The parameter value specified in the connect string takes precedence over the value specified in sqlnet.ora or tnsnames.ora.

When using the AUTHENTICATION_SERVICE value ALL (the default value), the server attempts to authenticate using each of the following methods. The server falls back to the authentication methods that appear further down on the list if attempts to use the authentication methods appearing higher on the list were unsuccessful. When using local database password authentication (no external authentication), set AUTHENTICATION_SERVICE=(NONE) for better client performance.

- Authentication based on a service external to the database, such as a service on the network layer, Kerberos, or RADIUS.
- Authentication based on the operating system user's membership in an administrative operating system group. Group names are platform-specific. This authentication applies to administrative connections only.
- Authentication performed by the database.
- Authentication based on credentials stored in a directory server.

Operating system authentication enables access to the database using any user name and any password when an administrative connection is attempted, such as using the AS SYSDBA clause when connecting using SQL*Plus. An example of a connection is as follows.

```
sqlplus ignored_username/ignored_password AS SYSDBA
```

When the operating-system user who issued the preceding command is already a member of the appropriate administrative operating system group, then the connection is successful. This is because the user name and password are ignored by the server because Oracle checks the group membership first.

Default

ALL

> **Note:**
>
> When installing Oracle Database with Database Configuration Assistant (DBCA), you can set this parameter to NTS in the sqlnet.ora file.

Values

Authentication methods that are available with Oracle Net Services:

- NONE for no authentication methods, including Microsoft Windows native operating system authentication. When you set AUTHENTICATION_SERVICE to NONE, then the user can use a valid user name and password to access the database.
- ALL for all authentication methods.
- BEQ for native operating system authentication for operating systems other than Microsoft Windows.
- KERBEROS5 for Kerberos authentication.

- `NTS` for Microsoft Windows native operating system authentication. In this case, the user must authenticate to the database with OS credentials using Windows native authentication. No external password is needed. NTS checks the group membership for an OS user. For example, if an OS user is a member of the `ORA_DBA` group, then the user can log in to the database as `SYSDBA`.

  > **Note:**
  >
  > With the `AUTHENTICATION_SERVICE=NTS` setting, if you try to connect through SQL*Plus using NTS authentication and specify an external password (for example, `SQL*Plus SYSTEM/password`), then the connection fails with an `ORA-12638, 00000, "Failed to retrieve credentials for adapter_name.` error message. For regular user name and password based authentication, set the value to `NONE`.

- `RADIUS` for Remote Authentication Dial-In User Service (RADIUS) authentication.
- `TCPS` for TLS authentication.

Example

```
net_service_name=
    (DESCRIPTION=
        (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
        (SECURITY=(AUTHENTICATION_SERVICE=(KERBEROS5)))
        (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
    )
```

Related Topics

- *Oracle Database Security Guide*
- SQLNET.AUTHENTICATION_SERVICES
  Use the `sqlnet.ora` parameter `SQLNET.AUTHENTICATION_SERVICES` to enable one or more authentication services.

6.10.2 AZURE_DB_APP_ID_URI

Use the `AZURE_DB_APP_ID_URI` parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).

Purpose

To specify the unique app ID URI of the database instance registered with Azure AD. This is the protected resource identifier (on Azure AD) for which the database client application requests an access token during token-based authentication.

Usage Notes

- You must set this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` authentication flows.

- This URI value is used to compose the authorization scope (permission) of your database token request:

  ```
  $Scope = "database_app_id_uri/scope"
  ```

  For example:

  ```
  $Scope = "https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3/session:scope:connect"
  ```

  In this example, `https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3` is the app ID URI and `session:scope:connect` is the scope.
- For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the app ID URI value by logging in to the Azure portal. This is listed as the Application ID URI value on the App registrations - Overview page of the Azure Active Directory service.

Specify the app ID URI in the following format:

Azure_AD_tenancy_url/application_(client)_id

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
```

```
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In these examples, the optional `CLIENT_ID`, `TENANT_ID`, and `REDIRECT_URI` parameters are not specified. Thus, the client automatically gets these values from the Azure SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

6.10.3 CLIENT_CERTIFICATE

Use the `CLIENT_CERTIFICATE` parameter to specify the file system path to a client certificate that authenticates the database client.

Purpose

File system path to a client certificate that authenticates the database client application registered with Microsoft Azure Active Directory (Azure AD). A client certificate is the digital certificate of an Azure cloud resource, and the client uses this certificate as a credential to prove its identity when requesting an Azure AD access token. This is used for the `AZURE_SERVICE_PRINCIPAL` token-based authentication flow.

Usage Notes

This is an optional parameter. When a client secret is not configured, the client driver reads the file system path of a client certificate from the `AZURE_CLIENT_CERTIFICATE_PATH` environment variable in the Azure SDK configuration. You can use this parameter along with the `TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL` setting to override the default certificate path. Note that this parameter is ignored if the client driver is configured with a client secret.

If you have not configured the SDKs, then you must set this parameter (along with other required parameters, such as `CLIENT_ID` and `TENANT_ID`). Otherwise, an error message appears prompting you to configure all required parameters.

For all supported clients (JDBC-thin clients, ODP.NET Core classes, and ODP.NET Managed Driver classes), you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

Full path (including a file name) to the Azure certificate file

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (CLIENT_CERTIFICATE=ORACLE_HOME/.azure/certificates/my-
app.pem))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&CLIENT_CERTIFICATE=ORACLE_HOME/.azure/certificates/my-
app.pem
```

In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

6.10.4 CLIENT_ID

Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify the ID of the Azure AD application. This is the unique application (client) ID assigned to your application by Azure AD when the application is registered. This application is your database client that requests to get an access token for the user during Azure AD token-based authentication.

Usage Notes

You use this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` authentication flows.

This is an optional parameter. You can set it in these scenarios:

- For the AZURE_MANAGED_IDENTITY authentication flow (applicable to client-side or server-side applications hosted on Azure environments, such as Azure App Service or Azure virtual machine), the client driver uses a system-assigned managed identity. A system-assigned managed identity is an implicit identity assigned by Azure AD to your application, and is configured in the Azure SDK by default.

  You can use this parameter to explicitly assign the client ID of a user-assigned managed identity to your application.
- For all other authentication flows, if you have configured the Azure SDKs, then the client driver automatically searches for the client ID in the SDK configuration. If you have not configured the SDKs, then you must set this parameter (along with other required parameters, such as TENANT_ID and CLIENT_CERTIFICATE). Otherwise, an error message appears prompting you to configure all required parameters.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or tnsnames.ora connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the sqlnet.ora file, Easy Connect syntax, or tnsnames.ora connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the client ID value by logging in to the Azure portal. This is listed as the Application (client) ID value on the App registrations - Overview page.

Examples

In the tnsnames.ora file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the sqlnet.ora file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3
CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

In these examples, the optional `TENANT_ID` and `REDIRECT_URI` parameters are not specified. The client driver automatically gets these values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

6.10.5 IGNORE_ANO_ENCRYPTION_FOR_TCPS

The `IGNORE_ANO_ENCRYPTION_FOR_TCPS` parameter specifies if the `SQLNET.ENCRYPTION_CLIENT` parameter should be ignored for this specific TNS alias.

Purpose

To specify if the `SQLNET.ENCRYPTION_CLIENT` parameter should be ignored for this specific TNS alias.

Usage Notes

If your requirements are that `SQLNET.ENCRYPTION_SERVER` be set to `required`, then you can set the `IGNORE_ANO_ENCRYPTION_FOR_TCPS` parameter in both `SQLNET.ENCRYPTION_CLIENT` and `SQLNET.ENCRYPTION_SERVER` to `TRUE`. This forces the client to ignore the value that is set for the `SQLNET.ENCRYPTION_CLIENT` parameter for all outgoing TCPS connections.

Default

`FALSE`

Example 6-12 Example

```
test_ssl=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=)(PORT=1750))
       (CONNECT_DATA=(SID=^ORACLE_SID^))
       (SECURITY=(IGNORE_ANO_ENCRYPTION_FOR_TCPS=TRUE))
     )
```

6.10.6 KERBEROS5_CC_NAME

Use the `tnsnames.ora` parameter `KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.

Purpose

To specify the complete path name to the Kerberos CC file.

Usage Notes

- In addition to the `tnsnames.ora` file or connect string, you can set this parameter in the `sqlnet.ora` file. Note that this parameter is called `SQLNET.KERBEROS5_CC_NAME` in the `sqlnet.ora` file. The connect string value takes precedence.
- This parameter supports multiple principals for the storage of credentials that are returned by KDC in an encrypted format.

  You can use the `okinit`, `oklist`, and `okdstry` utilities to configure encrypted cache files for all Kerberos principals. These utilities work with encrypted cache files only if you specify the cache path using `KERBEROS5_CC_NAME`.
- `KERBEROS5_CC_NAME` is mandatory for all additional Kerberos users and principals. Optionally, you can set the `KERBEROS5_PRINCIPAL` parameter to specify the Kerberos principal name associated with the credential cache (specified through `KERBEROS5_CC_NAME`). You can set `KERBEROS5_PRINCIPAL` in the connect string, `sqlnet.ora` file, or `tnsnames.ora` file.

  Oracle Database checks `KERBEROS5_PRINCIPAL` against the value that is retrieved from the credential cache. If the two values do not match, then the user is not authenticated.

Values and Examples

You can use the following formats to specify a value for `KERBEROS5_CC_NAME`:

- If the Oracle database is using a directory cache:
  - `KERBEROS5_CC_NAME=`*`complete_path_to_cc_file`*

    For example:

    ```
    KERBEROS5_CC_NAME=/tmp/kcache
    ```

    ```
    KERBEROS5_CC_NAME=D:\tmp\kcache
    ```
  - `KERBEROS5_CC_NAME=FILE:`*`complete_path_to_cc_ file`*

    For example:

    ```
    KERBEROS5_CC_NAME=FILE:/tmp/kcache
    ```
- If the Oracle database is using the native Windows cache:
  - `KERBEROS5_CC_NAME=OSMSFT://`
  - `KERBEROS5_CC_NAME=MSLSA:`

  The `OSMSFT` and `MSLSA` options specify that the file is on Microsoft Windows and is running Microsoft Kerberos Key Distribution Center (KDC).

Default

The default value is operating system-dependent, as follows:

- On Linux and UNIX operating systems: `/tmp/krb5cc_userid`
- On Microsoft Windows operating systems: `c:\tmp\krbcache`

Related Topics

- KERBEROS5_PRINCIPAL
  Use the `KERBEROS5_PRINCIPAL` parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.

- SQLNET.KERBEROS5_CC_NAME
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- *Oracle Database Security Guide*

6.10.7 KERBEROS5_PRINCIPAL

Use the `KERBEROS5_PRINCIPAL` parameter to set the Kerberos principal name associated with the Kerberos credentials cache (CC) file.

Purpose

When you configure Kerberos authentication for an Oracle Database client, you can specify multiple Kerberos principals with a single Oracle Database client.

This is an optional parameter. When specified, it is used to verify if the principal name in the credential cache (specified using `KERBEROS5_CC_NAME`) matches the parameter value.

Usage Notes

Use this parameter in the `SECURITY` section of the `tnsnames.ora` file, or set it in the `sqlnet.ora` file. Alternatively, you can set `KERBEROS5_PRINCIPAL` in the connect string along with the `KERBEROS5_CC_NAME` parameter to connect as a different Kerberos principal.

The parameter value specified in the connect string takes precedence over the value specified in the `sqlnet.ora` or `tnsnames.ora` file.

Each Kerberos principal must have a valid credential cache. Oracle Database checks `KERBEROS5_PRINCIPAL` against the value that is retrieved from the credential cache. If the two values do not match, then the user is not authenticated.

Examples

- For a user `krbuser1`, who is externally authenticated using the Kerberos principal `krbprinc1@example.com` and the credential cache for this principal is located at `/tmp/krbuser1/krb.cc`, the connect descriptor in the `tnsnames.ora` file is:

```
net_service_name=
(DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-svr)(PORT=1521))
      (CONNECT_DATA=(SERVICE_NAME=sales.example.com))
      (SECURITY=
          (KERBEROS5_CC_NAME=/tmp/krbuser1/krb.cc)
          (KERBEROS5_PRINCIPAL=krbprinc1@example.com)))
```

  In the `sqlnet.ora` file:

```
SQLNET.KERBEROS5_CC_NAME=/tmp/krbuser1/krb.cc
KERBEROS5_PRINCIPAL=krbprinc1@example.com
```

- For a user `krbuser2`, who is externally authenticated using the Kerberos principal `krbprinc2@example.com` and the credential cache for this principal is located at `/tmp/krbuser2/krb.cc`, the connect descriptor in the `tnsnames.ora` file is:

```
net_service_name=
(DESCRIPTION=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-svr)(PORT=1521))
      (CONNECT_DATA=(SERVICE_NAME=sales.example.com))
      (SECURITY=
          (KERBEROS5_CC_NAME=/tmp/krbuser2/krb.cc)
          (KERBEROS5_PRINCIPAL=krbprinc2@example.com)))
```

  In `sqlnet.ora` file:

```
SQLNET.KERBEROS5_CC_NAME=/tmp/krbuser2/krb.cc
KERBEROS5_PRINCIPAL=krbprinc2@example.com
```

> **Note:**
>
> The connection fails if the principal in the `/tmp/krbuser1/krb.cc` file does not contain the `krbprinc1@example.com` value.

Related Topics

- SQLNET.KERBEROS5_CC_NAME
  Use the `sqlnet.ora` parameter `SQLNET.KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- KERBEROS5_CC_NAME
  Use the `tnsnames.ora` parameter `KERBEROS5_CC_NAME` to specify the complete path name to the Kerberos credentials cache (CC) file.
- *Oracle Database Security Guide*

6.10.8 OCI_COMPARTMENT

Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.

Purpose

To define the scope of your database token request. This value instructs the database client to initiate a token request to databases within the specified compartment only. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Usage Notes

The `OCI_COMPARTMENT` parameter is optional if you have not specified the `OCI_DATABASE` parameter. If you choose to set `OCI_DATABASE`, then you must also set `OCI_COMPARTMENT` to limit your token request to the specified database within that compartment.

If you do not set both the OCI_COMPARTMENT and OCI_DATABASE parameters, then the entire tenancy is the scope of your token request.

You can use this parameter along with the PASSWORD_AUTH and TOKEN_AUTH authentication settings:

- With the PASSWORD_AUTH configuration, the database client can only request an IAM database token using the IAM user name and IAM database password.
- With the TOKEN_AUTH configuration, the database client can request an IAM database token using the API-key, delegation token, security token, resource principal, or instance principal credentials. You can also enable the database client to directly retrieve the db-token with IAM Single-Sign On (SSO) credentials by using the OCI_INTERACTIVE, OCI_API_KEY, OCI_INSTANCE_PRINCIPAL, OCI_DELEGATION_TOKEN, and OCI_RESOURCE_PRINCIPAL authentication flows.

Use this parameter under the SECURITY section of the tnsnames.ora file, sqlnet.ora file, Easy Connect syntax, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCID for the IAM compartment to allow access for the database token. You can get the OCID value for your compartment from the Compartments information page in the OCI console.

The compartment OCID uses this syntax:

OCI_COMPARTMENT=*compartment_OCID*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the tnsnames.ora file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345)
        (OCI_COMPARTMENT=ocid1.compartment..12345)
        (OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
OCI_COMPARTMENT=ocid1.compartment..12345
OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

In the Easy Connect syntax:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_COMPARTMENT=ocid1.compartment..12345&OCI_DATAB
ASE=ocid1.autonomousdatabase.oc1.12345
```

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH
- TOKEN_AUTH
- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

6.10.9 OCI_CONFIG_FILE

Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.

Purpose

To specify the directory location of the OCI configuration file. This file stores the client connection information for OCI Identity and Access Management (IAM) users as part of their profile. The SDK, CLI, and other OCI tools use this file to access the IAM user credentials during IAM token-based authentication.

Usage Notes

This is an optional parameter. If you do not set this parameter, then the database client gets the user's profile from the default configuration file located at `C:/user-profile/.oci/config`. You can use this parameter to override the default configuration file location. In this case, the database client searches for the profile in the location specified by `OCI_CONFIG_FILE`.

You can use this parameter along with the `TOKEN_AUTH` parameter for the `OCI_API_KEY` and `OCI_INTERACTIVE` authentication flows:

- When using the `OCI_INTERACTIVE` authentication flow, if this parameter is not set and the configuration file is also not present in the default location, then Oracle Database prompts the user for a region ID, presenting a list of region IDs from which the user can choose.
- When using the `OCI_API_KEY` authentication flow, if this parameter is not set and the default configuration file is also not present, then an ORA-50109 error message is

returned. In this case, you must set this parameter to include the configuration file location.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

Full path (including a file name) to the OCI configuration file

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=OCI_INTERACTIVE)
          (OCI_CONFIG_FILE=/home/dbuser1/config))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=OCI_INTERACTIVE
OCI_CONFIG_FILE=/home/dbuser1/config
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_CONFIG_FILE=/home/dbuser1/config
```

In these examples, the optional `OCI_PROFILE` parameter is not specified. Thus, the client automatically gets the `DEFAULT` profile from the specified configuration file directory.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

6.10.10 OCI_DATABASE

Use the OCI_DATABASE parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

Purpose

To define the scope of your database token request. The database OCID value instructs the database client to initiate a token request to the specified database within your compartment. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Usage Notes

This is an optional parameter. You can set this parameter to limit the access to only a particular database. If you set OCI_DATABASE, then you must also provide specific compartment identifier using the OCI_COMPARTMENT parameter.

You can use this parameter along with the PASSWORD_AUTH and TOKEN_AUTH authentication settings:

- With the PASSWORD_AUTH configuration, the database client can only request an IAM database token using the IAM user name and IAM database password.
- With the TOKEN_AUTH configuration, the database client can request an IAM database token using the API-key, delegation token, security token, resource principal, or instance principal credentials. You can also enable the database client to directly retrieve the db-token with IAM Single-Sign On (SSO) credentials by using the OCI_INTERACTIVE, OCI_API_KEY, OCI_INSTANCE_PRINCIPAL, OCI_DELEGATION_TOKEN, and OCI_RESOURCE_PRINCIPAL authentication flows.

Specify this parameter under the SECURITY section of the tnsnames.ora file, sqlnet.ora file, Easy Connect syntax, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

OCID of the database that you want to access for the client connection. You can get the OCID value for your database from the Database details page in the OCI console.

The database OCID uses this syntax:

OCI_DATABASE=*database_OCID*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the tnsnames.ora file:

```
net_service_name=
  (DESCRIPTION=
```

```
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345)
        (OCI_COMPARTMENT=ocid1.compartment..12345)
        (OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
OCI_COMPARTMENT=ocid1.compartment..12345
OCI_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

In the Easy Connect syntax:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_COMPARTMENT=ocid1.compartment..12345&OCI
_DATABASE=ocid1.autonomousdatabase.oc1.12345
```

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH
- TOKEN_AUTH
- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.

6.10.11 OCI_IAM_URL

Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

Purpose

To specify the IAM URL for your REST API requests. The database client connects to this URL to retrieve the database token from IAM.

Usage Notes

You set the `OCI_IAM_URL` parameter along with the `PASSWORD_AUTH` and `OCI_TENANCY` parameters while configuring IAM token-based authentication (using the IAM user name and IAM database password to retrieve the database token). These parameters are mandatory.

With this configuration, the database client can only request an IAM database token using the IAM user name and IAM database password. The client cannot request an IAM database token for an API-key, delegation token, security token, resource principal, service principal, or instance principal.

You can also set the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters to specify the scope of your token request.

Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCI IAM endpoint URL that the database client must connect with to get the database token. This URL is specific to your region and uses this syntax:

<authentication_regional_endpoint>/v1/actions/generateScopedAccessBearerToken

You can derive this value by replacing *<authentication_regional_endpoint>* with the API endpoint URL for your region. To obtain the appropriate API endpoint URL, see Identity and Access Management Data Plane API.

For example, if you want to use the URL as `https://auth.us-region-1.example.com`, then your OCI_IAM_URL value is:

```
https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
```

Examples

In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345))
```

```
    (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
)
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
```

In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH

6.10.12 OCI_PROFILE

Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

Purpose

To specify the profile name for IAM users. This profile is the client connection information stored in the OCI configuration file, and is used during IAM token-based authentication.

Usage Notes

This is an optional parameter. A profile named `DEFAULT` is already set in the configuration file. The database client gets the `DEFAULT` profile from the OCI configuration file (from either the default `C:/user-profile/.oci/config` directory location or the location specified by `OCI_CONFIG_FILE`). You can specify this parameter to override the `DEFAULT` profile set in the configuration file and assign a new profile name for the IAM user.

You use this parameter along with the `TOKEN_AUTH` parameter for the `OCI_API_KEY` and `OCI_INTERACTIVE` authentication flows.

When this parameter is not set and the profile is also not present in the configuration file, then an error message appears indicating that token-based authentication has failed due to the profile not existing.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Values

- `DEFAULT`: This means that no value is explicitly defined for a given profile, and the profile is inherited from the default profile set in the configuration file.
- *`profile_name`*: Specify a new configuration profile name (for example, `ADMIN_USER`) to override the `DEFAULT` profile set in the configuration file.

Default

```
DEFAULT
```

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=OCI_INTERACTIVE)
          (OCI_CONFIG_FILE=/home/dbuser1/config))
          (OCI_PROFILE=ADMIN_USER))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=OCI_INTERACTIVE
OCI_CONFIG_FILE=/home/dbuser1/config
OCI_PROFILE=ADMIN_USER
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=OCI_INTERACTIVE&OCI_CONFIG_FILE=/home/dbuser1/
config&OCI_PROFILE=ADMIN_USER
```

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH
- OCI_CONFIG_FILE
  Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.

6.10.13 OCI_TENANCY

Use the OCI_TENANCY parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.

Purpose

To specify OCID of the user's tenancy (root compartment).

Usage Notes

You set this parameter along with the mandatory PASSWORD_AUTH and OCI_IAM_URL parameters while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).

With this configuration, the database client can only request an IAM database token using the IAM user name and IAM database password. The client cannot request an IAM database token for an API-key, delegation token, security token, resource principal, service principal, or instance principal.

You can also set the optional OCI_COMPARTMENT and OCI_DATABASE parameters to specify the scope of your token request. If you do not set the OCI_COMPARTMENT and OCI_DATABASE parameter values, then the entire tenancy is the scope of your token request.

Use this parameter under the SECURITY section of the tnsnames.ora file, sqlnet.ora file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Default

None

Value

OCID of the user's tenancy. You can get the OCID value for your tenancy from the Tenancy information page in the OCI console.

The tenancy OCID uses this syntax:

OCI_TENANCY=*tenancy_OCID*

For details on the syntax options, see Oracle Cloud IDs (OCIDs).

Examples

In the tnsnames.ora file:

```
net_service_name=
  (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))
     (SECURITY=
        (SSL_SERVER_DN_MATCH=TRUE)
        (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
        (PASSWORD_AUTH=OCI_TOKEN)
        (OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
```

```
generateScopedAccessBearerToken)
        (OCI_TENANCY=ocid1.tenancy..12345))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
PASSWORD_AUTH=OCI_TOKEN
OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/
generateScopedAccessBearerToken
OCI_TENANCY=ocid1.tenancy..12345
```

In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- PASSWORD_AUTH

6.10.14 PASSWORD_AUTH

Use the `PASSWORD_AUTH` parameter to configure an authentication method for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS). With this setting, client connections use the IAM user name and IAM database password for logging in users to the database.

Purpose

To configure either IAM database password verifier authentication or IAM token-based authentication, using the IAM user name and IAM database password for the access.

For password verifier authentication, the database server retrieves an IAM database password verifier from IAM. For token-based authentication, the database client requests a database token (`db-token`) from IAM.

Usage Notes

- Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.
- This setting instructs the database client to either use the existing password login process with the database server (password verifier authentication) or to get a token with the IAM user name and IAM database password (token-based authentication). This IAM database password is different from the OCI console password. An IAM user can set this password from the OCI console.

  See Create an OCI IAM password to use for Autonomous Databases User Authentication and Authorization.
- By default, this parameter is set to `PASSWORD_VERIFIER`. The `PASSWORD_AUTH=PASSWORD_VERIFIER` setting configures IAM database password verifier authentication. The database server retrieves an IAM database password verifier (an encrypted hash of password) from IAM to authenticate users.

When an IAM user logs in with the IAM user name and IAM database password using `@connect_identifier`, the `PASSWORD_AUTH=PASSWORD_VERIFIER` setting along with `@connect_identifier` instructs the database client to follow the existing user name and password login process with the database server.

You can use the `PASSWORD_AUTH` parameter to override the `tnsnames.ora` or `sqlnet.ora` setting by specifying a different value in the connect string.

- To configure IAM token-based authentication with the IAM user name and IAM database password, set `PASSWORD_AUTH=OCI_TOKEN`. The database client requests a database token (`db-token`) from IAM for the user to access the database.

  This `db-token` obtained by the client is a bearer token with an expiration time and scope, and does not come with a private key. These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.

  When an IAM user logs in with the IAM user name and IAM database password using `/@connect_identifier`, the `PASSWORD_AUTH=OCI_TOKEN` setting along with `/@connect_identifier` instructs the database client to get the token directly from an OCI IAM endpoint using a REST API request. If the IAM user is mapped to a database schema (exclusively or shared), then the login is completed.

  For the database client to retrieve the token from IAM, you must set additional parameters so that the database client can find the IAM endpoint along with additional meta-data. The additional parameters are `OCI_IAM_URL` and `OCI_TENANCY` along with the optional `OCI_COMPARTMENT` and `OCI_DATABASE`. These values enable the database client to make appropriate calls to the specified endpoint.

  The `OCI_IAM_URL` parameter specifies the API endpoint URL that the database client must connect with. The `OCI_TENANCY` parameter specifies the OCID (Oracle Cloud Identifier) of the user's tenancy. The optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters limit the scope of your request.

  This authentication method is more secure than using a password verifier because a password verifier is considered sensitive. Also, only the database client can retrieve the database token. Applications or tools cannot pass these types of tokens through the database client API.

> **Note:**
>
> You can also use other IAM user credentials (such as API-key, security token, resource principal, service principal, instance principal, or delegation token) to get the `db-token`. This `db-token` is a proof-of-possession (PoP) token. In this case, you use a different parameter setting (`TOKEN_AUTH=OCI_TOKEN`).
>
> Unlike the IAM database password that can only be used by the database client to retrieve the token, these credentials require an application or tool to retrieve the token. See TOKEN_AUTH.

Default

PASSWORD_VERIFIER

Values and Examples

Value	Example
For IAM database password verifier authentication: PASSWORD_AUTH=PASSWORD_VERIFIER **Note**: Use of IAM user name and IAM database password with the IAM database password verifier is the default configuration, and you do not need to set any additional parameters for the client. However, if PASSWORD_AUTH is set to OCI_TOKEN in the client-side sqlnet.ora file, then the client tries to connect with OCI IAM to retrieve a database token using the IAM user name and IAM database password. In this case, you can override this setting for a particular connection using PASSWORD_AUTH=PASSWORD_VERIFIER.	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps) (HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext") (PASSWORD_AUTH=PASSWORD_VERIFIER)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: PASSWORD_AUTH=PASSWORD_VERIFIER

Value	Example

For IAM token-based authentication with the IAM user name and IAM database password:
`PASSWORD_AUTH=OCI_TOKEN`
Note: You must configure the TCPS protocol (`PROTOCOL=tcps`) and set the `SSL_SERVER_DN_MATCH` parameter to `TRUE` for token-based authentication.
	In the `tnsnames.ora` file:
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(PASSWORD_AUTH=OCI_TOKEN)`
	`(OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/generateScopedAccessBearerToken)`
	`(OCI_TENANCY=ocid1.tenancy..12345))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	In the `sqlnet.ora` file:
	`SSL_SERVER_DN_MATCH=TRUE`
	`PASSWORD_AUTH=OCI_TOKEN`
	`OCI_IAM_URL=https://auth.us-region-1.example.com/v1/actions/generateScopedAccessBearerToken`
	`OCI_TENANCY=ocid1.tenancy..12345`
	In these examples, the optional `OCI_COMPARTMENT` and `OCI_DATABASE` parameters are not specified and thus the entire tenancy is set as the scope of the token request.

Related Topics

- *Oracle Database Security Guide*
- OCI_IAM_URL
  Use the `OCI_IAM_URL` parameter to specify an endpoint URL that the database client must connect with to get the database token for authenticating Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users on OCI Database as a Service (DBaaS).
- OCI_TENANCY
  Use the `OCI_TENANCY` parameter to specify Oracle Cloud Identifier (OCID) of the user's tenancy.
- OCI_COMPARTMENT
  Use the `OCI_COMPARTMENT` parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.

- OCI_DATABASE
  Use the `OCI_DATABASE` parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.

6.10.15 REDIRECT_URI

Use the `REDIRECT_URI` parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify the redirect URI (or reply URL) registered for the Azure AD application. This redirect URI is the client application address where the authorization server sends the authentication response after successfully authenticating the user. This is used for the `AZURE_INTERACTIVE` token-based authentication flow.

Usage Notes

This is an optional parameter. If you do not set this parameter, then the database client reads the redirect URI value from the Azure SDK configuration. You can use this parameter along with the `TOKEN_AUTH=AZURE_INTERACTIVE` setting to override the default location. The authorization server redirects the user to this location only if you have registered the redirect URI for the client application in the Azure portal.

For all supported clients (JDBC-thin clients, ODP.NET Core classes, and ODP.NET Managed Driver classes), you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the redirect URI by logging in to the Azure portal. These URIs are listed as Redirect URIs on the App registrations - Authentication page of the Azure Active Directory service.

Specify the redirect URI in the following format:

```
http://localhost
```

The URI may also include a port number as follows:

```
http://localhost:port
```

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
```

```
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (REDIRECT_URI=http://localhost:1575))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&REDIRECT_URI=http://localhost:1575
```

In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically retrieves the client and tenant ID values from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

6.10.16 SECURITY

Use the `SECURITY` parameter to change the security properties of a connection.

Purpose

To change the security properties of a connection.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter. `SECURITY` permits additional parameters as listed in Security Section.

Example

```
net_service_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com))
    (SECURITY=
      (SSL_SERVER_CERT_DN="cn=sales,cn=OracleContext,dc=us,dc=acme,dc=com")))
```

6.10.17 SSL_CERTIFICATE_ALIAS

Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_ALIAS` to specify the alias of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.

Purpose

To specify the alias that you have provided when storing the client certificate in an Oracle Database wallet.

When encrypting mTLS connections between the database client and database server, the database client needs to provide a signed certificate to the database server. You can store this client certificate in an Oracle Database wallet or Microsoft Certificate Store (MCS). If there is more than one certificate that can be used, the user or application settings can specify the specific one to connect with. This choice can be made manually by the user via graphical user interface (GUI) or automatically by the application using a thumbprint or alias name. A thumbprint or alias name can uniquely identify the client certificate.

This parameter instructs the client to automatically select a particular certificate using the specified alias name. Thus, the user does not need to manually select the correct client certificate from the list available in a wallet.

Usage Notes

Use this parameter in the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

`orapki` helps you manage certificates and wallets for Oracle Database. To get the alias name value, run the following command:

```
orapki wallet display -wallet <wallet directory> -pwd <wallet password> -
complete
```

Value

Certificate alias name

Default

None

Examples

- In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
    (SECURITY=(SSL_CERTIFICATE_ALIAS=my_cert))
  )
```

- In the Easy Connect string:

```
tcps://salesserver:1521/sales.us.example.com?SSL_CERTIFICATE_ALIAS=my_cert
```

- In the `sqlnet.ora` file:

```
SSL_CERTIFICATE_ALIAS=my_cert
```

Related Topics

- *Oracle Database Security Guide*

6.10.18 SSL_CERTIFICATE_THUMBPRINT

Use the `sqlnet.ora` or `tnsnames.ora` parameter `SSL_CERTIFICATE_THUMBPRINT` to specify the thumbprint of the client certificate, to use in a Mutual Transport Layer Security (mTLS) connection.

Purpose

To specify the thumbprint signature for an X509 certificate. These thumbprints are automatically generated for certificates.

When encrypting mTLS connections between the database client and database server, the database client needs to provide a signed certificate to the database server. You can store this client certificate in an Oracle Database wallet or Microsoft Certificate Store (MCS). If there is more than one certificate that can be used, the user or application settings can specify the specific one to connect with. This choice can be made manually by the user via graphical user interface (GUI) or automatically by the application using a thumbprint or alias name. A thumbprint or alias name can uniquely identify the client certificate.

This parameter instructs the client to automatically select a particular certificate using the specified thumbprint. Thus, the user does not need to manually select the correct client certificate from the list available in a certificate store.

Usage Notes

Use this parameter in the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

You can specify both the SHA-1 and SHA-256 thumbprint information for the client certificate.

`orapki` helps you manage certificates and wallets for Oracle Database. To get the thumbprint value, run the following command:

```
orapki wallet display -wallet <wallet directory> -pwd <wallet
password> -complete
```

Value

SHA-1 or SHA-256 thumbprint of the client certificate, in the *<Algorithm>*:*<Hash>* format

For example:

```
SHA1:1B:11:01:5A:B1:2C:20:B2:12:34:3E:04:7B:83:47:DE:70:2E:4E:11


SHA256:B3:8A:5B:1A:03:63:83:92:2B:5D:E1:53:61:EE:03:94:0A:56:B4:56:41:7E:41:2
4:41:9B:88:EB:C6:1E:11:23
```

or

```
SHA1:1B11015AB12C20B212343E047B8347DE702E4E11


SHA256:B38A5B1A036383922B5DE15361EE03940A56B456417E4124419B88EBC61E1123
```

Default

None

Examples

- In the `tnsnames.ora` file:

```
net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))

(SECURITY=(SSL_CERTIFICATE_THUMBPRINT=SHA1:1B:11:01:5A:B1:2C:20:B2:12:34:3
E:04:7B:83:47:DE:70:2E:4E:11))
  )


net_service_name=
  (DESCRIPTION=
    (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))

(SECURITY=(SSL_CERTIFICATE_THUMBPRINT=SHA1:1B11015AB12C20B212343E047B8347D
E702E4E11))
  )
```

- In the Easy Connect string:

```
tcps://salesserver:1521/sales.us.example.com?
SSL_CERTIFICATE_THUMBPRINT=SHA1:1B11015AB12C20B212343E047B8347DE702E4E11
```

- In the `sqlnet.ora` file:

```
SSL_CERTIFICATE_THUMBPRINT=SHA256:B38A5B1A036383922B5DE15361EE03940A56B456
417E4124419B88EBC61E1123
```

Related Topics

- *Oracle Database Security Guide*

6.10.19 SSL_CLIENT_AUTHENTICATION

Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).

Purpose

To enable client authentication in a TLS connection. The connection can be one-way or two-way (mutual TLS or mTLS).

Usage Notes

When set to `TRUE`, a two-way TLS connection is initiated. Both the client and server (including the listener) authenticate each other. For example, if you set this parameter to `TRUE` in the server configuration (server-side `sqlnet.ora`), then the server attempts to authenticate the client. If you set it to `TRUE` in the listener configuration (`listener.ora`), then the listener attempts to authenticate the client.

When set to `FALSE`, only the client authenticates the server and listener as a one-way TLS connection. For example, if you set this parameter to `FALSE` in the server configuration, then the server does not authenticate the client. If you set it to `FALSE` in the listener configuration, then the listener does not authenticate the client.

When set to `OPTIONAL`, the server behaves as follows:

- If the client sends a certificate, then the connection is completed as a two-way TLS connection after authenticating the client.
- If the client does not send a certificate, then the connection is completed as a one-way TLS connection.

Ensure that this parameter setting is consistent for the server or listener (on one side) and the client (on the other). Otherwise, the connection may fail. For example, if you enable client authentication in the server or listener configuration, then you must enable it in the client configuration.

Default

`TRUE`

Values

- `TRUE` | `ON` | `YES` | `1`: To enable mTLS
- `FALSE` | `OFF` | `NO` | `0`: To enable one-way TLS
- `OPTIONAL`: To enable both TLS and mTLS

Example

```
SSL_CLIENT_AUTHENTICATION=FALSE
```

Related Topics

- *Oracle Database Security Guide*

6.10.20 SSL_SERVER_CERT_DN

Use the SSL_SERVER_CERT_DN parameter to specify the distinguished name (DN) of the database server.

Purpose

To specify the DN of the database server for DN matching. DN matching adds another client-side check on the listener and server certificates to ensure that the certificates are the correct ones that the client expects.

Usage Notes

- Set this parameter in the tnsnames.ora file or connect string to enforce full DN matching. For partial DN matching, do not include this parameter.
- The value in the SSL_SERVER_CERT_DN parameter is matched to the full DN in the listener and server certificates only when the SSL_SERVER_DN_MATCH parameter is set to ON.

  The client must know the server DN ahead of time to specify the value in SSL_SERVER_CERT_DN. The client uses SSL_SERVER_CERT_DN to check the certificate DN for both the listener and database server that the client wants to connect to.
- Oracle recommends that you use the same certificate for both the listener and server. If you use different certificates with different DNs for the listener and server, then full DN matching fails. In this case, you need to get new certificates with the same DN (for full DN matching) or you need to change your DN matching strategy.
- Starting with Oracle Database 23ai, the DN matching behavior is enhanced for better security. Full DN matching checks the complete DN in SSL_SERVER_CERT_DN against the DN in both the listener and server certificates. In earlier releases, full DN matching checked the complete DN only in the server certificate.

  If you want to revert to the earlier weaker DN matching behavior (that is, checking only the server certificate), then set SSL_ALLOW_WEAK_DN_MATCH=TRUE. However, note that the SSL_ALLOW_WEAK_DN_MATCH parameter is deprecated and will be removed in a future release. Oracle recommends that you get new certificates or change your DN matching strategy.

Value

Database server DN

Example

```
finance=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL = tcps) (HOST = finance)
     (PORT = 1575)))
   (CONNECT_DATA=
     (SERVICE_NAME=finance.us.example.com))
   (SECURITY=
     (SSL_SERVER_CERT_DN="cn=finance,cn=OracleContext,c=us,o=example")))
```

Related Topics

- SSL_SERVER_DN_MATCH
  Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.
- SSL_ALLOW_WEAK_DN_MATCH
  Use the `sqlnet.ora` parameter `SSL_ALLOW_WEAK_DN_MATCH` to allow the earlier weaker distinguished name (DN) matching behavior during server-side certificate validation.
- *Oracle Database Security Guide*

6.10.21 SSL_SERVER_DN_MATCH

Use the `SSL_SERVER_DN_MATCH` parameter to enforce server-side certificate validation through distinguished name (DN) matching.

Purpose

To enforce server-side certificate validation through DN matching.

The purpose of adding this DN matching parameter for the client is to further improve security on a Transport Layer Security (TLS) connection. A TLS connection relies on the client to verify if the database server certificate is valid and signed by a trusted root certificate. The listener and server certificate DN matching adds another client-side check on the listener and server certificates to ensure that the certificates are the correct ones that the client expects.

Usage Notes

- If you set this parameter to `TRUE`, then in addition to verifying the server's certificate chain, the client enforces another check against the listener and server through DN matching.
- You can configure either partial DN matching or full DN matching.

  Through partial DN matching, the client checks the `HOSTNAME` parameter (in the client `sqlnet.ora` file or connect string) against a host name in the certificate DN or certificate Subject Alternative Name (SAN) field. The client checks `HOSTNAME` against both the listener and server certificates in this order:

  1. The client first compares `HOSTNAME` with a host name in the listener certificate's DN. For example, CN part of DN:

     ```
     "c=us,o=examplecorporation,cn=sales.us.example.com"
     ```

  2. If no match is found in the listener certificate's DN, then the client compares `HOSTNAME` with a host name in the listener certificate's SAN field. For example:

     ```
     "DNS:sales.us.example.com"
     ```

     If no match is found in the listener certificate's SAN field, then the client does not try connecting to the server and the connection fails.
  3. If the listener certificate check succeeds, then the client performs similar checks on the server certificate. That is, the client first compares `HOSTNAME` with a host name in the server certificate's DN.

4. If no match is found in the server certificate's DN, then the client compares `HOSTNAME` with a host name in the server certificate's SAN field.

Through full DN matching, the client checks the complete DN in `SSL_SERVER_CERT_DN` against the certificate DN of both the listener and server certificates. To enforce a full DN match, specify the complete DN using the `SSL_SERVER_CERT_DN` parameter in the `tnsnames.ora` file or connect string.

- Oracle recommends that you use the same certificate for both the listener and server.

  If you use different certificates with different DNs for the listener and server, then full DN matching fails. In this case, you need to get new certificates with the same DN (for full DN matching) or you need to change your DN matching strategy. If you have configured partial DN matching, then it may also fail if `HOSTNAME` is not found in the certificate DN or SAN fields of both the listener and server certificates.

- Prior to Oracle Database 23ai, partial DN matching checked against host name and SAN only in the server certificate. If a match was not found, then along with the host name and SAN, it also checked the `SERVICE_NAME` parameter. Similarly, full DN matching checked against the complete DN only in the server certificate.

  If you want to revert to the earlier weaker DN matching behavior (that is, checking only the server certificate and allowing a service name check for partial DN matching), then set `SSL_ALLOW_WEAK_DN_MATCH=TRUE`. However, note that the `SSL_ALLOW_WEAK_DN_MATCH` parameter is deprecated and will be removed in a future release. Oracle recommends that you get new certificates or change your DN matching strategy.

Default

`NO`

Values

- `YES` | `ON` | `TRUE` | `1`:

  To enforce partial or full DN matching. If the DN matches the host name or SAN in both the listener and server certificates, then the connection succeeds. If the DN does not match the host name or SAN in the server or listener certificate, then the connection fails.

- `NO` | `OFF` | `FALSE` | `0`:

  To not enforce DN matching. If the DN does not match the host name or SAN in the sever or listener certificate, then the connection is successful, but an error is logged to the `sqlnet.log` file.

Example

```
SSL_SERVER_DN_MATCH=YES
```

Related Topics

- SSL_SERVER_CERT_DN
  Use the `SSL_SERVER_CERT_DN` parameter to specify the distinguished name (DN) of the database server.
- SSL_ALLOW_WEAK_DN_MATCH
  Use the `sqlnet.ora` parameter `SSL_ALLOW_WEAK_DN_MATCH` to allow the earlier weaker distinguished name (DN) matching behavior during server-side certificate validation.
- *Oracle Database Security Guide*

6.10.22 SSL_VERSION

Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.

Purpose

To define the version of TLS that must run on the systems with which the database server communicates. By default, the database server and client negotiate the strongest security protocol. Oracle does not recommend modifying this parameter, unless your security requirements mandate the usage of certain protocol versions.

Usage Notes

- Clients, listeners, and database servers must use compatible versions. Modify this parameter only when necessary to enforce the use of the more secure TLS protocol and not allow clients that only work with the older TLS protocols. The current default uses TLS 1.3, which is the version required for multiple security compliance requirements. If you need to specify TLS 1.2, then also include TLS 1.3 to allow more secure connections.
- In addition to `sqlnet.ora`, `listener.ora`, and `cman.ora`, you can specify this parameter under the `SECURITY` section of `tnsnames.ora` or directly as part of the connect string. The parameter value specified in the connect string takes precedence over the other specified values.
- Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

  In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.
- Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.

  SSLv3 is a much less secure protocol to secure the database server-to-client connection. Instead of using SSLv3, allow the database server and client to negotiate the most secure protocol that is common between the server and the client. Oracle Database 23ai provides TLS 1.2 and TLS 1.3 protocols for certificate-based network encryption.
- If you set `SSL_VERSION` to `undetermined`, then the most secure TLS protocol version is used. You can also use the `SSL_VERSION=undetermined` setting in the connect string for a specific connection to override the `SSL_VERSION` value configured in the `sqlnet.ora`, `listener.ora`, or `cman.ora` file.
- If you do not set `SSL_VERSION` to any value, then all the supported TLS protocol versions are tried starting with the most secure version. This is typically the most common configuration, ensuring that the strongest protocol is chosen during TLS negotiation.

Values

`undetermined | TLSv1.2 | TLSv1.3`

Default

`undetermined`

Syntax and Examples

- To specify a single protocol version:

  `SSL_VERSION=`*`TLS_protocol_version`*

  For example:

  `SSL_VERSION=TLSv1.3`

- To specify multiple protocol versions, use a comma-separated string of values, enclosed in parenthesis:

  `SSL_VERSION=(`*`TLS_protocol_version1`*`,`*`TLS_protocol_version2`*`)`

  For example:

  `SSL_VERSION=(TLSv1.2,TLSv1.3)`

> **Note:**
>
> Do not enclose protocol versions in parenthesis while specifying this parameter in the `tnsnames.ora` file or as part of the connect string, otherwise the setting will not parse correctly. For example:
>
> ```
> net_service_name=
>   (DESCRIPTION=
>      (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver)(PORT=1522))
>      (SECURITY=(SSL_VERSION=TLSv1.2,TLSv1.3))
>   )
> ```

Related Topics

- Set the Required TLS Version on the Server
- Set the Required TLS Version on the Client

6.10.23 TENANT_ID

Use the `TENANT_ID` parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.

Purpose

To specify the ID of the Azure AD tenant in which your Azure AD application is registered. This is the unique tenant ID that identifies your database instance in Azure AD.

Usage Notes

You use this parameter along with the `TOKEN_AUTH` parameter for the `AZURE_INTERACTIVE`, `AZURE_SERVICE_PRINCIPAL`, `AZURE_MANAGED_IDENTITY`, and `AZURE_DEVICE_CODE` token-based authentication flows.

This is an optional parameter. If you have configured the Azure SDKs, then the client driver automatically searches for the tenant ID in the SDK configuration. If you have not configured the SDKs, then you must set this parameter (along with other required parameters, such as `CLIENT_ID` and `CLIENT_CERTIFICATE`). Otherwise, an error message appears prompting you to configure all required parameters.

For JDBC-thin clients, you can specify this parameter in the Easy Connect syntax or `tnsnames.ora` connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can specify this parameter in the `sqlnet.ora` file, Easy Connect syntax, or `tnsnames.ora` connect string. The parameter value specified in the connect string takes precedence.

Default

None

Value

You can get the tenant ID value by logging in to the Azure portal. This is listed as Tenant ID on the Tenant Properties page.

Examples

In the `tnsnames.ora` file:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))
       (SECURITY=
          (SSL_SERVER_DN_MATCH=TRUE)
          (SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")
          (TOKEN_AUTH=AZURE_INTERACTIVE)
          (AZURE_DB_APP_ID_URI=https://application.example.com/
123ab4cd-1a2b-1234-a12b-aa00123b2cd3)
          (TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123)
          (REDIRECT_URI=http://localhost:1575))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

In the `sqlnet.ora` file:

```
SSL_SERVER_DN_MATCH=TRUE
TOKEN_AUTH=AZURE_INTERACTIVE
AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3
TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123
REDIRECT_URI=http://localhost:1575
```

In the Easy Connect string:

```
tcps:sales-svr:1521/sales.us.example.com?
TOKEN_AUTH=AZURE_INTERACTIVE&AZURE_DB_APP_ID_URI=https://
application.example.com/123ab4cd-1a2b-1234-a12b-
aa00123b2cd3&TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-
a12bcdab0123&REDIRECT_URI=http://localhost:1575
```

In these examples, the optional `CLIENT_ID` parameter is not specified. Thus, the client automatically gets the client ID value from the SDK configuration.

Related Topics

- *Oracle Database Security Guide*
- TOKEN_AUTH

6.10.24 TOKEN_AUTH

Use the `TOKEN_AUTH` parameter to configure token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) or Microsoft Azure Active Directory (Azure AD) users. With this setting, the database client looks for a token file when a `/` (slash) login is used.

Purpose

Token-based access enforces strong authentication, which enables a more secure access to the database. IAM users can connect to OCI Database as a Service (DBaaS) databases, and Azure AD users can connect to Oracle Databases (cloud or on-premises).

Use this parameter under the `SECURITY` section of the `tnsnames.ora` file, `sqlnet.ora` file, or directly as part of the command-line connect string. The parameter value specified in the connect string takes precedence over the other specified values.

Usage Notes for IAM

- An OCI IAM token (`db-token`), which is obtained from IAM using Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI) or programmatically from the OCI Software Development Kit (SDK), is a proof-of-possession (PoP) token with an expiration time and scope.

  You can use one of the IAM user credentials, such as API-key, security token, resource principal, instance principal, or delegation token to retrieve the `db-token` and private key from IAM.

- These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.
- You must configure the TCPS protocol (`PROTOCOL=tcps`) and set the `SSL_SERVER_DN_MATCH` parameter to `TRUE` for token-based authentication.
- When an IAM user logs in using `/@connect_identifier` (and `TOKEN_AUTH` is set to `OCI_TOKEN`), the `TOKEN_AUTH=OCI_TOKEN` setting along with `/@connect_identifier` instructs the database client to get the `db-token` and private key from either the default directory or the location specified by `TOKEN_LOCATION`.
- If your client application is updated to retrieve tokens from IAM, then you can override the `TOKEN_AUTH=OCI_TOKEN` setting. The client application gets the `db-token` and private key from IAM and sends as attributes to the database client using the client API. In this case, you do not need to specify the `TOKEN_AUTH` and `TOKEN_LOCATION` parameters.
- The general IAM token-based authentication process is as follows:

  1. An IAM user or application in OCI first requests the `db-token` from IAM by using API-key, security token, resource principal, service principal, instance principal, or delegation token (delegation token is available only in the Cloud Shell).

     To use a security token, you need to generate it by completing the browser authentication process and then request the `db-token` using that security token. If the IAM policy that authorizes you to be issued the `db-token` exists, then the `db-token` is returned.

     You request the `db-token` using OCI CLI (or OCI SDK for applications). For example, run the following OCI CLI command to request the `db-token` by using an API-key (`apikey`):

     ```
     $ oci iam db-token get --profile scott
     ```

     The `profile` option specifies the profile for which you want to access the IAM user credentials and retrieve the `db-token`.

     For more information on using OCI CLI, see the `get` command details in *Oracle Cloud Infrastructure CLI Command Reference*.

  2. OCI CLI references the `config` file (that stores your IAM user credentials as part of the profile) and makes a call to IAM to get the `db-token`. The `db-token` and private key files are written at the default or specified token location.
  3. You can specify the `TOKEN_LOCATION` parameter to override the default directory where the `db-token` and private key files are stored.

     The database client gets the `db-token` and private key from the default token location or the location specified by `TOKEN_LOCATION`, signs the `db-token` with the private key and sends it to the database server. The database server verifies the `db-token` and gets the group membership information for the user. If the IAM user is mapped to a database schema (exclusively or shared), then the login is completed.

- The following authentication flows enable the database client to directly retrieve the `db-token` with IAM Single-Sign On (SSO) credentials.

Note that this feature is available in environments that use JDBC-thin clients, ODP.NET Core classes, or ODP.NET Managed Driver classes. For JDBC-thin clients, you can set this in the `tnsnames.ora` or Easy Connect connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can set this in the `sqlnet.ora`, `tnsnames.ora`, or Easy Connect connect string. The parameter value specified in the connect string takes precedence. To configure this feature for JDBC-thin clients, see *Oracle Database JDBC Developer's Guide* and for ODP.NET, see *Oracle Data Provider for .NET Developer's Guide*.

- `TOKEN_AUTH=OCI_INTERACTIVE` specifies the OCI Interactive flow. This authenticates the token request interactively using a web browser, and is useful for client-side web applications or desktop applications.

  The database client gets a default profile (named `DEFAULT`) from the OCI configuration file, which is stored either in the default directory or at the location specified by the `OCI_CONFIG_FILE` parameter. After validating the user's region against a list of valid regions, the client launches an authentication request to the user in a web browser, prompting to log in using the IAM user name and password along with any additional factors required by IAM.

  Optionally, you can override the `DEFAULT` profile set in the configuration file by specifying the `OCI_PROFILE` parameter.

- `TOKEN_AUTH=OCI_API_KEY` specifies the OCI API Key flow. This authenticates the token request with IAM using an IAM-recognized API-key.

  The database client reads the file system location of the API-key from the user's `DEFAULT` profile in the OCI configuration file, from either the default configuration file directory or the location specified by `OCI_CONFIG_FILE`.

  Optionally, you can override the user's `DEFAULT` profile set in the configuration file by specifying the `OCI_PROFILE` parameter.

- `TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL` specifies the OCI Instance Principal flow. This authenticates the token request with IAM as an OCI instance principal for applications running on OCI compute instances.

- `TOKEN_AUTH=OCI_DELEGATION_TOKEN` specifies the OCI Delegation Token flow. This authenticates the token request with IAM using a delegation token for applications running in an OCI Cloud Shell.

- `TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL` specifies the OCI Resource Principal flow. This authenticates the token request with IAM as an OCI resource principal for applications running in a container (as an OCI function).

- `TOKEN_AUTH=OCI_DEFAULT` specifies the Default flow. With this setting, the client driver reads the predefined environment variables from the SDK configuration, evaluates each authentication flow in a sequence, and then assigns the most appropriate flow based on the environment where the application is running.

  This is the sequence in which the driver evaluates each authentication flow with `OCI_DEFAULT`:

  1. OCI API Key: The driver first checks if a configuration file is present at the location specified by the `OCI_CONFIG_FILE` parameter or at the default location (`$HOME/.oci/config`). The driver then checks if the file contains a profile matching the name configured by the `OCI_PROFILE` parameter or the default name (`DEFAULT`). Finally, the driver checks if the profile is configured with an entry named `key_file`. If all of these checks succeed, then authentication with an API key is used. If any of these checks fail, then the driver proceeds to the next step.

2. OCI Delegation Token: The driver first checks if the `OCI_CONFIG_FILE` environment variable is set. The driver then checks if a file is present at the location configured by the `OCI_CONFIG_FILE` environment variable. The driver then checks if the file contains a profile named `DEFAULT`. Finally, the driver checks if the profile is configured with an entry named `delegation_token_file`. If all of these checks succeed, then authentication with a delegation token is used. If any of these checks fail, then the driver proceeds to the next step.
3. OCI Resource Principal: The driver first checks if the `OCI_RESOURCE_PRINCIPAL_VERSION` environment variable is set. The driver then checks if the variable is set to version `2.2` or `1.1`. If the variable is set to `2.2`, the driver then checks if the `OCI_RESOURCE_PRINCIPAL_PRIVATE_PEM`, `OCI_RESOURCE_PRINCIPAL_RPST`, and `OCI_RESOURCE_PRINCIPAL_REGION` environment variables are also set. Otherwise, if the variable is set to `1.1`, then the driver checks if the `OCI_RESOURCE_PRINCIPAL_RPT_ENDPOINT` environment variable is also set. If the required variables for a version are set, then authentication as a resource principal is used. If any variable is not set, then the driver proceeds to the next step.
4. OCI Instance Principal: The driver requests a certificate from the instance metadata service. The base URL of the service is `http://169.254.169.254/opc/v2/`. However, a fallback URL of `http://169.254.169.254/opc/v1/` is used if the `v2` service request fails. If a request to the `v2` or `v1` service succeeds, then authentication as an instance principal is used. If the request fails, then the driver proceeds to the next step.
5. The driver reports an error indicating that authentication is not possible using any of the authentication flows.

You also need to specify the `OCI_DATABASE` and `OCI_COMPARTMENT` parameters for all these authentication flows, if the OCI database token policy limits you to access only a particular database or databases within a compartment.

> **Note:**
>
> You can also use another IAM credential, IAM database password, to request the `db-token` from IAM. This `db-token` is a bearer token and does not come with a private key. You can configure the database client to request this token using your IAM user name and IAM database password. An application cannot pass this type of `db-token` to the client. In this case, you use a different parameter setting (`PASSWORD_AUTH=OCI_TOKEN`).
>
> Unlike the API-key, security token, resource principal, service principal, instance principal, and delegation token that require an application or tool to get a token, the IAM database password can only be used by the database client to retrieve the token. See PASSWORD_AUTH.

Default Setting for IAM

None

Table 6-1 Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_TOKEN`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=OCI_TOKEN))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_TOKEN`
	
	In these examples, the optional `TOKEN_LOCATION` parameter is not specified. Thus, the client automatically gets the `db-token` and private key from the default token location.

Table 6-1 (Cont.) Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_INTERACTIVE`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=OCI_INTERACTIVE))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_INTERACTIVE`
	
	In these examples, the optional `OCI_CONFIG` and `OCI_PROFILE` parameters are not specified. Thus, the client automatically gets the `DEFAULT` profile from the default configuration file directory.

Table 6-1 (Cont.) Values and Examples for IAM

Value	Example

`TOKEN_AUTH=OCI_API_KEY`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=OCI_API_KEY))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_API_KEY`
	
	In these examples, the optional `OCI_CONFIG` and `OCI_PROFILE` parameters are not specified. Thus, the client automatically gets the API-key value from the `DEFAULT` profile stored in the default configuration file directory.

Table 6-1 (Cont.) Values and Examples for IAM

Value	Example
TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context") (TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_INSTANCE_PRINCIPAL
TOKEN_AUTH=OCI_DELEGATION_TOKEN	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context") (TOKEN_AUTH=OCI_DELEGATION_TOKEN)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_DELEGATION_TOKEN

Table 6-1 (Cont.) Values and Examples for IAM

Value	Example
TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context") (TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=OCI_RESOURCE_PRINCIPAL

Usage Notes for Azure AD

- An Azure AD OAuth2 access token is a bearer token with an expiration time and scope. This token follows the OAuth2.0 standard with Azure AD extensions. You can request these tokens from tools and scripts run on Linux, Microsoft PowerShell, or other environments. You can also request these tokens programmatically using the Microsoft SDKs.
- These tokens are transmitted over secure channels. You must use only the TCP/IP with Transport Layer Security (TLS) protocol, otherwise an error message appears indicating that non-TLS connections are disallowed.
- You must configure the TCPS protocol (PROTOCOL=tcps) and set the SSL_SERVER_DN_MATCH parameter to TRUE for token-based authentication.
- When an Azure AD user logs in using */@connect_identifier* (and TOKEN_AUTH is set to OAUTH), the TOKEN_AUTH=OAUTH setting instructs the database client to get the access token from the directory location specified by TOKEN_LOCATION if the token file is named token. If the token file name is different from token, then you must use the file name along with the directory location while specifying the TOKEN_LOCATION parameter.

  The TOKEN_LOCATION parameter is mandatory for Azure AD token-based authentication. The database client gets the token from this location and sends it to the database server.
- If your client application is updated to retrieve tokens from Azure AD, then you can override the TOKEN_AUTH=OAUTH setting. Azure AD directly passes the db-token as an

attribute to the database client using the client API. When the client receives this request, the client sends it to the database server.

In this case, you do no need to specify the `TOKEN_AUTH` and `TOKEN_LOCATION` parameters.

- The general Azure AD token-based authentication process is as follows:

  1. An Azure AD user or application first requests the access token from Azure AD using one of the supported Microsoft Azure AD authentication flows (resource owner password credentials, authorization code, on-behalf-of (OBO) flow, or client credentials).

     An Azure AD user can connect using any supported utility to retrieve the token and store it in a local file directory.

     You can request the token from tools and scripts run on Linux, Microsoft PowerShell, or other environments. You can also request programmatically using the Microsoft SDKs.
     For detailed examples on how to retrieve an Azure AD OAuth2 access token, see *Oracle Database Security Guide*.

  2. The database client then sends the token to the database server. The database server verifies the token (using the Azure AD public key) and extracts various claims from the token, including user name, app roles, and audience. If the Azure AD principal is mapped to a database schema (exclusively or shared), then the login is completed.

- The following authentication flows enable the database client to directly retrieve an access token with Azure AD SSO credentials.

  Note that this feature is available in environments that use JDBC-thin clients, ODP.NET Core classes, or ODP.NET Managed Driver classes. For JDBC-thin clients, you can set this in the `tnsnames.ora` or Easy Connect connect string. For ODP.NET Core classes and ODP.NET Managed Driver classes, you can set this in the `tnsnames.ora` connect string or `sqlnet.ora` file (except `REDIRECT_URI` and `CLIENT_CERTIFICATE`). The parameter value specified in the connect string takes precedence. To configure this feature for JDBC-thin clients, see *Oracle Database JDBC Developer's Guide* and for ODP.NET, see *Oracle Data Provider for .NET Developer's Guide*.

  – `TOKEN_AUTH=AZURE_INTERACTIVE` specifies the Azure OAuth2 Interactive flow. This authenticates the token request interactively using a web browser, and is useful for client-side web applications or desktop applications.

    The database client launches an authentication request to the user (either in a dialog box if the user is using a web application or as a prompt if the user is working in a command line shell), prompting to log in using the Azure AD user name and password. After logging in to the Azure AD account, the user is redirected back to the client application (to its registered redirect URI).

    You must set the `AZURE_DB_APP_ID_URI` and `REDIRECT_URI` parameters to compose the scope URL and redirect URI (or reply URL) for your Azure AD application. The client driver reads the client ID and tenant ID values from the Azure SDK configuration. If required, you can set the `CLIENT_ID` and `TENANT_ID` parameters to override the default values.

  – `TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL` specifies the Azure Service Principal flow. This authenticates the token request as a service principal by using either

a client secret or a client certificate, and is useful for server-side applications (for example, microservices or back-end apps).

If the client driver is not configured with a client secret, then the client driver reads the file system location of the Azure certificate from the `AZURE_CLIENT_CERTIFICATE_PATH` environment variable in the Azure SDK configuration.

You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. The client driver reads the client ID, tenant ID, and client certificate path from the Azure SDK configuration. If required, you can set the `CLIENT_ID`, `TENANT_ID`, and `CLIENT_CERTIFICATE` parameters to override the default values.

- `TOKEN_AUTH=AZURE_MANAGED_IDENTITY` specifies the Azure Managed Identity flow. This authenticates the token request with Azure AD as an Azure managed identity, and is useful for client-side or server-side applications hosted on Azure environments (for example, Azure App Service or Azure virtual machine).

  You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. By default, the client driver uses a system-assigned managed identity from the Azure SDK configuration. If required, you can set the `CLIENT_ID` parameter to configure a user-assigned managed identity for authenticating the token request.

- `TOKEN_AUTH=AZURE_DEVICE_CODE` specifies the Azure Device Code flow. This authenticates the token request interactively, and is useful for client-side applications running on platforms with limited or no browser support (for example, command line tool, such as SQLcl).

  The database client displays a device code and an Azure AD login URL through the standard output of the tool, and prompts the user to enter the device code, Azure AD user name, and Azure AD password on any browser-supporting device (for example, cellphone or laptop). After completing the login in a web browser, the Azure SDK returns an access token to the client. The client sends the access token to the database to authorize the database user session.

  You must set the `AZURE_DB_APP_ID_URI` parameter to compose the scope URL for your token request. The client driver reads the client ID and tenant ID values from the Azure SDK configuration. If required, you can set the `CLIENT_ID` and `TENANT_ID` parameters to override the default values.

  > **Note:**
  >
  > You must explicitly enable the Azure OAuth2 Interactive and Azure Device Code flows for your Azure AD app in the Azure portal. To do so, on the App registrations - Authentication page, under Advanced Settings, set **Allow public client flows** to **Yes**.

- `TOKEN_AUTH=AZURE_DEFAULT` specifies the Default flow. With this setting, the client driver reads the predefined environment variables from the SDK configuration, evaluates each authentication flow in a sequence, and then assigns the most appropriate flow based on the environment where the application is running.

  This is the sequence in which the driver evaluates each authentication flow with `AZURE_DEFAULT`:

1. Azure Service Principal with Client Secret Credentials: The driver checks if client ID and client secret are configured as parameters to the driver or as SDK environment variables. If both are configured, then the driver authenticates as a service principal using a client secret. Otherwise, the driver proceeds to the next step.
2. Azure Service Principal with Client Certificate Credentials: The driver checks if client ID and client certificate are configured as parameters to the driver or SDK environment variables. If both are configured, then the driver authenticates as a service principal using a client certificate. Otherwise, the driver proceeds to the next step.
3. Azure Username Credentials: The driver checks if client ID, username, and password are configured as parameters to the driver or SDK environment variables. If all are configured, then the driver authenticates as a service principal using the username and password. Otherwise, the driver proceeds to the next step.
4. Azure Managed Identity: The driver checks if the `MSI_ENDPOINT` or `IDENTITY_ENDPOINT` environment variable is set. If either is set, then the driver authenticates as a managed identity using the configured endpoint. If neither is set, then the driver checks if the `AZURE_TENANT_ID` and `AZURE_FEDERATED_TOKEN_FILE` environment variables are set. If both are set, then the driver authenticates as a managed identity using the configured token file. If both are not set, then the driver requests an access token from the Azure Instance Metadata Service (IMDS) endpoint. If the request succeeds, then the driver authenticates as a managed identity. Otherwise, the driver proceeds to the next step.
5. Visual Studio Credentials: For ODP.NET Core classes and ODP.NET Managed Driver classes, the driver additionally evaluates the Azure user through Visual Studio Credentials authentication flow. The driver checks if the `TENANT_ID` parameter or the `AZURE_TENANT_ID` environment variable is set and if the Azure user is logged in to Visual Studio. If both the checks succeed, then authentication with the Visual Studio credentials is used. Otherwise, the driver proceeds to the next step.
6. The driver reports an error indicating that authentication is not possible using any of the authentication flows.

Default Setting for Azure AD

None

Table 6-2 Values and Examples for Azure AD

Value	Example

`TOKEN_AUTH=OAUTH`	• If the token file is named `token`, `TOKEN_AUTH=OAUTH`, and `TOKEN_LOCATION="`*`token_file_directory`*`"`:
	In the `tnsnames.ora` file:
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Or`
	`acleContext")`
	`(TOKEN_AUTH=OAUTH)`
	`(TOKEN_LOCATION="/home/dbuser1/`
	`access-token"))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.exam`
	`ple.com))`
	`)`
	In the `sqlnet.ora` file:
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-`
	`token"`
	In these examples, the token file name is `token`. Thus, only the directory path (`/home/dbuser1/access-token`) is specified. The client automatically looks for the `token` file in the specified path and gets the access token.
	• If the token file name is different from `token`, `TOKEN_AUTH=OAUTH`, and `TOKEN_LOCATION="`*`token_file_directory/token_filename`*`"`:
	In the `tnsnames.ora` file:
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)`
	`(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Or`

Table 6-2 (Cont.) Values and Examples for Azure AD

Value	Example

	acleContext")
	(TOKEN_AUTH=OAUTH)
	(TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"))
	
	(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
	)
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"`
	
	In these examples, the token file name is `mytoken`. Thus, both the file name and directory path (`/home/dbuser1/access-token`) are specified. The client gets the access token from the `mytoken` file in the specified path.

Table 6-2 (Cont.) Values and Examples for Azure AD

Value	Example

`TOKEN_AUTH=AZURE_INTERACTIVE`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=AZURE_INTERACTIVE)`
	`(AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3)`
	`(REDIRECT_URI=http://localhost:1575))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=AZURE_INTERACTIVE`
	`AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3`
	`REDIRECT_URI=http://localhost:1575`
	
	In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Table 6-2 (Cont.) Values and Examples for Azure AD

Value	Example
TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL	In the tnsnames.ora file: net_service_name= (DESCRIPTION = (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521)) (SECURITY= (SSL_SERVER_DN_MATCH=TRUE) (SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context") (TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL) (AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3)) (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))) In the sqlnet.ora file: SSL_SERVER_DN_MATCH=TRUE TOKEN_AUTH=AZURE_SERVICE_PRINCIPAL AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3 In these examples, the optional CLIENT_ID, TENANT_ID, and CLIENT_CERTIFICATE parameters are not specified. Thus, the client automatically gets the required values from the SDK configuration.

Table 6-2 (Cont.) Values and Examples for Azure AD

Value	Example

`TOKEN_AUTH=AZURE_MANAGED_IDENTITY`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=AZURE_MANAGED_IDENTITY)`
	`(AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=AZURE_MANAGED_IDENTITY`
	`AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3`
	
	In these examples, the optional `CLIENT_ID` parameter is not specified. The client driver uses a system-assigned managed identity to authenticate the token request with Azure AD.

Table 6-2 (Cont.) Values and Examples for Azure AD

Value	Example

`TOKEN_AUTH=AZURE_DEVICE_CODE`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=Oracle Context")`
	`(TOKEN_AUTH=AZURE_DEVICE_CODE)`
	`(AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=AZURE_DEVICE_CODE`
	`AZURE_DB_APP_ID_URI=https://application.example.com/123ab4cd-1a2b-1234-a12b-aa00123b2cd3`
	
	In these examples, the optional `CLIENT_ID` and `TENANT_ID` parameters are not specified. Thus, the client automatically gets the client ID and tenant ID values from the SDK configuration.

Related Topics

- Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
- Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases
- TOKEN_LOCATION
  Use the `TOKEN_LOCATION` parameter to specify the directory location where token file is stored for token-based authentication.
- OCI_CONFIG_FILE
  Use the `OCI_CONFIG_FILE` parameter to specify the directory location where the Oracle Cloud Infrastructure (OCI) configuration file is stored.
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile name for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) users.

- OCI_DATABASE
  Use the OCI_DATABASE parameter to specify Oracle Cloud Identifier (OCID) of the database that you want to access for the client connection.
- OCI_COMPARTMENT
  Use the OCI_COMPARTMENT parameter to specify Oracle Cloud Identifier (OCID) of the compartment that holds database instances for client connections.
- AZURE_DB_APP_ID_URI
  Use the AZURE_DB_APP_ID_URI parameter to specify the app ID URI of the Oracle Database instance registered with Microsoft Azure Active Directory (Azure AD).
- REDIRECT_URI
  Use the REDIRECT_URI parameter to specify the redirect URI registered for the Microsoft Azure Active Directory (Azure AD) application.
- CLIENT_ID
  Use the CLIENT_ID parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.
- TENANT_ID
  Use the TENANT_ID parameter to specify the ID of your Microsoft Azure Active Directory (Azure AD) tenant.
- CLIENT_CERTIFICATE
  Use the CLIENT_CERTIFICATE parameter to specify the file system path to a client certificate that authenticates the database client.

6.10.25 TOKEN_LOCATION

Use the TOKEN_LOCATION parameter to specify the directory location where token file is stored for token-based authentication.

Purpose

To specify the token file directory location. You use this parameter while configuring token-based authentication for Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) or Microsoft Azure Active Directory (Azure AD) users. The database client gets the token from this location and sends it to the database server. For Azure AD, you can also specify the token file name along with the directory location.

Use this parameter along with the TOKEN_AUTH parameter in the tnsnames.ora file, sqlnet.ora file, or directly as part of the command-line connect string. The parameter values specified in the connect string take precedence over the other specified values.

Usage Notes for IAM

The TOKEN_LOCATION parameter is optional for IAM token-based authentication. You can use this parameter along with the TOKEN_AUTH parameter to override the default directory where the db-token and private key are stored. This location is used by the database client to retrieve the db-token and private key.

When an IAM user initiates a connection using /@*connect_identifier* (and TOKEN_AUTH is set to OCI_TOKEN), the database client retrieves the db-token and private key from either the default directory or the location specified by TOKEN_LOCATION. The client then signs the db-token using the private key and sends the db-token to the database server.

Default Setting for IAM

- On Linux:

  `/home/username/.oci/db-token`

- On Windows:

  The database client searches for the default directory in this order:

  If the `USERPROFILE` environment variable is set, then the client searches in the `USERPROFILE` directory (for example, `C:\Users\username`).

  If `USERPROFILE` is not set, then the client searches in `HOMEDRIVE` directory (for example, `C:`) with `HOMEPATH` (for example, `\Users\username`).

  For example, the default token location directory on Windows is:

  `C:\Users\username\.oci\db-token`

Values and Examples for IAM

Value	Example

`TOKEN_LOCATION="token_file_directory"`	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION =`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1521))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OCI_TOKEN)`
	`(TOKEN_LOCATION="/home/oracle/.oci/db-token"))`
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OCI_TOKEN`
	`TOKEN_LOCATION="/home/oracle/.oci/db-token"`

Usage Notes for Azure AD

The `TOKEN_LOCATION` parameter is mandatory for Azure AD token-based authentication. You must use this parameter along with the `TOKEN_AUTH` parameter to specify the directory location where the Azure AD OAuth2 access token is stored. This location is used by the database client to get the access token.

If your token file is named `token`, then specify only the directory path. If the token file name is different from `token`, then you must use the file name along with the directory path.

When an Azure AD user initiates a connection using `/@connect_identifier`, the database client retrieves the access token from the location specified by `TOKEN_LOCATION` and sends the token to the database server.

Default Setting for Azure AD

None

Values and Examples for Azure AD

Value	Example

If the token file is named `token`:
`TOKEN_LOCATION="token_file_directory"`
	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=TRUE)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OAUTH)(TOKEN_LOCATION="/home/dbuser1/access-token"))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token"`
	
	In these examples, the token file name is `token`. Thus, only the directory path (`/home/dbuser1/access-token`) is specified. The client automatically looks for the `token` file in the specified path and gets the access token.

Value	Example

If the token file name is different from `token`:
`TOKEN_LOCATION="`*`token_file_directory/token_filename`*`"`
	In the `tnsnames.ora` file:
	
	`net_service_name=`
	`(DESCRIPTION=`
	`(ADDRESS=(PROTOCOL=tcps)(HOST=salesserver1)(PORT=1522))`
	`(SECURITY=`
	`(SSL_SERVER_DN_MATCH=ON)`
	
	`(SSL_SERVER_CERT_DN="C=US,O=example,CN=OracleContext")`
	`(TOKEN_AUTH=OAUTH)`
	`(TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"))`
	
	`(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))`
	`)`
	
	In the `sqlnet.ora` file:
	
	`SSL_SERVER_DN_MATCH=TRUE`
	`TOKEN_AUTH=OAUTH`
	`TOKEN_LOCATION="/home/dbuser1/access-token/mytoken"`
	
	In these examples, the token file name is `mytoken`. Thus, both the file name and directory path (`/home/dbuser1/access-token`) are specified. The client gets the access token from the `mytoken` file in the specified path.

Related Topics

- Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
- Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases
- TOKEN_AUTH

6.10.26 WALLET_LOCATION

Use the `WALLET_LOCATION` parameter in the `tnsnames.ora` file to specify different locations where Oracle wallets are stored.

Purpose

This parameter denotes a connection specific wallet. You can use this parameter when different connections need to use different wallets on the client side.

Usage Notes

- The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

  For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.

- The connect string parameter `MY_WALLET_DIRECTORY` has been deprecated with Oracle Database 23ai.

  Oracle recommends that you use `WALLET_LOCATION` in the connect string to override the `sqlnet.ora WALLET_LOCATION` setting. `WALLET_LOCATION` has been updated for connect strings so that the same parameter can be used in `sqlnet.ora` and in `tnsnames.ora` This change simplifies the parameters that you need to remember. Oracle recommends that you change your client connect strings to use `WALLET_LOCATION` instead of `MY_WALLET_DIRECTORY`.

  Use of `WALLET_LOCATION` in `tnsnames.ora` overrides the `WALLET_LOCATION` in `sqlnet.ora` for the specific `tnsnames.ora` service. The `tnsnames.ora WALLET_LOCATION` setting enables a client connection to have distinct TLS connections that use certificates. This means that the client initiates multiple TLS connections using different TLS certificates, in the same client process.

- Use this parameter if you have a single client that must rely on more than one TLS session. An example would be for a client that requires access to multiple pluggable databases (PDBs), each with its own identity (certificate). This feature enables you to configure the client to connect to the correct identity for each PDB. After the configuration is complete, multi-threaded clients are able to access more than one wallet with different certificates in simultaneous TLS sessions.

Default

None

Examples

```
net_service_name=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
   (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com))
      (SECURITY=(wallet_location=/home/oracle/wallets/databases)))



ssl_certs1 =
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=shobeen.us.example.com) (PORT=1750))
       (CONNECT_DATA=(SID=sales_pdb))
       (SECURITY=(WALLET_LOCATION=/oracle/wallets/certificates/sales_cert))
     )
ssl_certs2 =
```

```
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=shobeen.us.example.com)
(PORT=1750))
       (CONNECT_DATA=(SID=marketing_pdb))
       (SECURITY=(WALLET_LOCATION=/oracle/wallets/certificates/
marketing_cert))
     )
```

Related Topics

- *Oracle Database Security Guide*
- *Oracle Database Net Services Reference*

6.11 Timeout Parameters

The timeout section of the `tnsnames.ora` file provides the ability to specify timeout and retry configuration through the TNS connect string.

The following parameters can be set at the `DESCRIPTION` level of a connect string:

- CONNECT_TIMEOUT
  Use the `tnsnames.ora` parameter `CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.
- RETRY_COUNT
  Use the `tnsnames.ora` parameter `RETRY_COUNT` to specify the number of times an `ADDRESS` list is traversed before terminating the connection attempt.
- RETRY_DELAY
  Use the `tnsnames.ora` parameter `RETRY_DELAY` to specify the delay between connection retries.
- TRANSPORT_CONNECT_TIMEOUT
  Use the `tnsnames.ora` parameter `TRANSPORT_CONNECT_TIMEOUT` to specify the transport connect timeout duration, in milliseconds, seconds, or minutes.
- RECV_TIMEOUT
  Use the `tnsnames.ora` parameter `RECV_TIMEOUT` to specify the duration of time that a database client or server should wait for data from a peer after establishing a connection.

6.11.1 CONNECT_TIMEOUT

Use the `tnsnames.ora` parameter `CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.

Purpose

To specify the timeout duration in `ms` or `msec` (milliseconds), `sec` (seconds), or `min` (minutes) for a client to establish an Oracle Net connection to Oracle Database.

Usage Notes

- Put this parameter under the `DESCRIPTION` parameter.

- In case, no unit is mentioned, the default unit is `sec`.

  It accepts different timeouts with or without space between the value and the unit. For example:

  ```
  CONNECT_TIMEOUT=10 ms
  ```

  or

  ```
  CONNECT_TIMEOUT=10ms
  ```

- The timeout interval specified by `CONNECT_TIMEOUT` is a superset of the TCP connect timeout interval. It includes the time to be connected to the database instance providing the requested service, not just the duration of the TCP connection.

  The timeout interval is applicable to each `ADDRESS` in an `ADDRESS_LIST`, and each IP address to which a host name is mapped.

- The `CONNECT_TIMEOUT` parameter is equivalent to the `sqlnet.ora` parameter `SQLNET.OUTBOUND_CONNECT_TIMEOUT`, and overrides it.

Examples

```
net_service_name=
 (DESCRIPTION=
  (CONNECT_TIMEOUT=10 ms)(RETRY_COUNT=3)
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
  (CONNECT_DATA=
   (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- SQLNET.OUTBOUND_CONNECT_TIMEOUT
  Use the `sqlnet.ora` parameter `SQLNET.OUTBOUND_CONNECT_TIMEOUT` to specify the amount of time, in milliseconds, seconds, or minutes, in which clients must establish Oracle Net connections to database instances.

6.11.2 RETRY_COUNT

Use the `tnsnames.ora` parameter `RETRY_COUNT` to specify the number of times an `ADDRESS` list is traversed before terminating the connection attempt.

Purpose

To specify the number of times an `ADDRESS` list is traversed before the connection attempt is terminated.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter.

When a `DESCRIPTION_LIST` is specified, each `DESCRIPTION` is traversed multiple times based on the specified number of retries.

Example

```
net_service_name=
(DESCRIPTION_LIST=
 (DESCRIPTION=
  (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1a-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1b-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales1.example.com)))
 (DESCRIPTION=
  (CONNECT_TIMEOUT=60)(RETRY_COUNT=1)
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2a-svr)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2b-svr)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales2.us.example.com))))
```

6.11.3 RETRY_DELAY

Use the `tnsnames.ora` parameter `RETRY_DELAY` to specify the delay between connection retries.

Purpose

To specify the delay between subsequent retries for a connection in units of `ms` or `msec` (milliseconds), `sec` (seconds), or `min` (minutes). This parameter works in conjunction with the `RETRY_COUNT` parameter.

Usage Notes

- Put this parameter under the `DESCRIPTION` parameter.

  When a `DESCRIPTION_LIST` is specified, each `DESCRIPTION` is traversed multiple times based on the specified number of retries, and the specific delay for the description.
- In case, no unit is mentioned, the default unit is `sec`.

  You can configure it with or without space between the value and the unit. For example:

  ```
  RETRY_DELAY=800 ms
  ```

  or

  ```
  RETRY_DELAY=800ms
  ```

Example

```
net_service_name=
(DESCRIPTION_LIST=
  (DESCRIPTION=
   (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)(RETRY_DELAY=800ms)
   (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=myhost1)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=myhost2)(PORT=1521)))
```

```
 (CONNECT_DATA=(SERVICE_NAME=example1.com)))
(DESCRIPTION=
 (CONNECT_TIMEOUT=60)(RETRY_COUNT=1)(RETRY_DELAY=5sec)
 (ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=tcp)(HOST=myhost3)(PORT=1521))
  (ADDRESS=(PROTOCOL=tcp)(HOST=myhost4)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME=example2.com))))
```

Related Topics

- RETRY_COUNT
  Use the `tnsnames.ora` parameter `RETRY_COUNT` to specify the number of times an `ADDRESS` list is traversed before terminating the connection attempt.

6.11.4 TRANSPORT_CONNECT_TIMEOUT

Use the `tnsnames.ora` parameter `TRANSPORT_CONNECT_TIMEOUT` to specify the transport connect timeout duration, in milliseconds, seconds, or minutes.

Purpose

To specify the transport connect timeout duration in `ms` or `msec` (milliseconds), `sec` (seconds), or `min` (minutes) for a client to establish an Oracle Net connection to Oracle Database.

Usage Notes

- This parameter is put under the `DESCRIPTION` parameter.
- The default value is 60 `sec`. In case, no unit is mentioned, the default unit is `sec`.

  It accepts different timeouts with or without space between the value and the unit.

  ```
  TRANSPORT_CONNECT_TIMEOUT=10 ms
  ```

  or

  ```
  TRANSPORT_CONNECT_TIMEOUT=10ms
  ```

- The timeout interval is applicable for each `ADDRESS` in an `ADDRESS_LIST` description, and each IP address that a host name is mapped. The `TRANSPORT_CONNECT_TIMEOUT` parameter is equivalent to the `sqlnet.ora` parameter `TCP.CONNECT_TIMEOUT`, and overrides it.

Example

```
net_service_name =
  (DESCRIPTION=
    (TRANSPORT_CONNECT_TIMEOUT=10 ms)
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
    (CONNECT_DATA=
      (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- TCP.CONNECT_TIMEOUT
  Use the `sqlnet.ora` parameter `TCP.CONNECT_TIMEOUT` to specify the amount of time in which a client must establish TCP connections to database servers.

6.11.5 RECV_TIMEOUT

Use the `tnsnames.ora` parameter `RECV_TIMEOUT` to specify the duration of time that a database client or server should wait for data from a peer after establishing a connection.

Purpose

To specify the time duration in `ms` or `msec` (milliseconds), `sec` (seconds), `min` (minutes), or `hr` (hours) for a database client or server to wait for data from the peer after establishing a connection. The peer must send data within the time interval that you specify.

Usage Notes

- This parameter is put under the `DESCRIPTION` parameter.
- If you do not specify a unit of measurement, then the default unit is `sec`.

  It accepts time duration with or without space between the value and the unit. For example:

  ```
  RECV_TIMEOUT=10 ms
  ```

  or

  ```
  RECV_TIMEOUT=10ms
  ```

- Setting this parameter for clients ensures that receive operations are not left in a wait state indefinitely or for a long period due to server host being down, server busy state, or network connectivity issues.

  If a client does not receive response data in the time specified, then the client logs `ORA-12535: TNS:operation timed out` and `ORA-12609: TNS: Receive timeout occurred` messages to the `sqlnet.log` file.

Default Value

None

Minimum Value

`1 ms`

Allowed Range

Any number greater than the minimum value of `1 ms` up to `4294967295 ms`.

Examples

```
net_service_name=
 (DESCRIPTION=
```

```
(CONNECT_TIMEOUT=10ms)(RETRY_COUNT=3)(RECV_TIMEOUT=10ms)
(ADDRESS_LIST=
 (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-svr)(PORT=1521))
 (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-svr)(PORT=1521)))
(CONNECT_DATA=
 (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

6.12 Compression Parameters

The compression section of the `tnsnames.ora` file provides the ability to enable compression and specify compression levels. These parameters can be set at the `DESCRIPTION` level of a connect string.

- COMPRESSION
  The `tnsnames.ora` file's compression parameter enables or disables the data compression.
- COMPRESSION_LEVELS
  The `COMPRESSION_LEVELS` parameter of the `tnsnames.ora` file specifies the compression level.

6.12.1 COMPRESSION

The `tnsnames.ora` file's compression parameter enables or disables the data compression.

Purpose

To enable or disable data compression.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter.

Setting this parameter in the connect descriptor for a client overrides the `SQLNET.COMPRESSION` parameter in the client-side `sqlnet.ora` file.

Default

`off`

Values

- `on` to enable data compression.
- `off` to disable data compression.

Example

```
net_service_name=
 (DESCRIPTION=
   (COMPRESSION=on)
      (ADDRESS_LIST=
         (ADDRESS= (PROTOCOL=tcp) (HOST=sales1-server) (PORT=1521))
```

```
      (ADDRESS= (PROTOCOL=tcp) (HOST=sales2-server) (PORT=1521)))
  (CONNECT_DATA=
      (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- SQLNET.COMPRESSION
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION` to enable or disable data compression.

6.12.2 COMPRESSION_LEVELS

The `COMPRESSION_LEVELS` parameter of the `tnsnames.ora` file specifies the compression level.

Purpose

To specify the compression level.

Usage Notes

The compression levels are used at the time of negotiation to verify which levels are used at both ends, and select one level. Put this parameter under the `DESCRIPTION` parameter.

This parameter is used with the `COMPRESSION` parameter. Setting this parameter in the connect descriptor for a client overrides the `SQLNET.COMPRESSION_LEVELS` parameter in the client-side `sqlnet.ora` file.

Default

`low`

Values

- `low` for low CPU usage and a low compression ratio.
- `high` for high CPU usage and a high compression ratio.

Example

```
net_service_name=
 (DESCRIPTION=
  (COMPRESSION=on)
  (COMPRESSION_LEVELS=(LEVEL=low)(LEVEL=high))
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

Related Topics

- SQLNET.COMPRESSION_LEVELS
  Use the `sqlnet.ora` parameter `SQLNET.COMPRESSION_LEVELS` to specify the compression level.

7

Centralized Configuration Provider Naming Parameters

This chapter describes the parameters for configuring the Centralized Configuration Provider naming method.

- Overview of the Centralized Configuration Provider Naming
  Azure App Configuration and Oracle Cloud Infrastructure (OCI) Object Storage provide services to centrally manage configuration data.
- Allowed Parameter Names and Values in Connect Descriptors
  These are the restrictions on the parameter names and values that can appear in the connect descriptor, when using the Centralized Configuration Provider naming method.
- Authentication Parameters for Azure App Configuration Store
  These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Azure App Configuration store.
- Authentication Parameters for OCI Object Storage
  These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.1 Overview of the Centralized Configuration Provider Naming

Azure App Configuration and Oracle Cloud Infrastructure (OCI) Object Storage provide services to centrally manage configuration data.

Instead of storing connect descriptors on a client in the `tnsnames.ora` file, you can store them in a Centralized Configuration Provider. This centralizes network service names and addresses in a single location, facilitating administration (addition, deletion, or modification) of connect descriptors. This also enables you to centrally manage password change policies for all stored database user names and passwords.

- **Azure App Configuration Store**:
  In the Azure App Configuration store, you organize connect descriptors and other details under a prefix. A Centralized Configuration Provider connect identifier takes this prefix from the Azure App Configuration endpoint and uses it to locate the connect descriptor, database user name and password, and other Oracle Call Interface configuration parameters.
- **JSON File in the OCI Object Storage**:

  In the OCI Object Storage, you organize connect descriptors and other details in a JSON file. A Centralized Configuration Provider connect identifier refers to JSON objects with specific names for resolving connect identifiers to connect descriptors.

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.2 Allowed Parameter Names and Values in Connect Descriptors

These are the restrictions on the parameter names and values that can appear in the connect descriptor, when using the Centralized Configuration Provider naming method.

- Naming-Restricted Parameter Names
  This is a list of all left-hand side naming parameters (before =) allowed in a `connect_descriptor` value, which you specify in the Azure App Configuration store or Oracle Cloud Infrastructure (OCI) Object Storage JSON file.
- Naming-Restricted Parameter Values
  For some of the naming-restricted parameter names, some values are not allowed on the right-hand side (after =).

7.2.1 Naming-Restricted Parameter Names

This is a list of all left-hand side naming parameters (before =) allowed in a `connect_descriptor` value, which you specify in the Azure App Configuration store or Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

All parameters that can potentially access the file-system are restricted to avoid errors, especially in multi-tenant, middle-tier environments. This is because a tenant may potentially add references to a wallet of another tenant. All the parameters that may cause process-level changes or issues with unintended resource usage are also restricted.

Parameter
ADDRESS
ADDRESS_LIST
BACKUP
COLOCATION_TAG
COMPRESSION
COMPRESSION_LEVELS
CONNECT_DATA
CONNECTION_ID_PREFIX
CONNECT_TIMEOUT
DELAY
DESCRIPTION
DESCRIPTION_LIST
ENABLE
EXPIRE_TIME
FAILOVER
FAILOVER_MODE

Parameter
GLOBAL_NAME
HOST
HS
HTTPS_PROXY
HTTPS_PROXY_PORT
IGNORE_ANO_ENCRYPTION_FOR_TCPS
INACTIVITY_TIMEOUT
INCREMENT
INSTANCE_NAME
LOAD_BALANCE
LOB_PREFETCH_SIZE
MAX
MAX_LIFETIME_SESSION
MAX_USE_SESSION
METHOD
MIN
OCI
POOL_BOUNDARY
POOL_CONNECTION_CLASS
POOL_NAME
POOL_PURITY
PORT
PREFETCH_ROWS
PROTOCOL
RDB_DATABASE
RECV_BUF_SIZE
RECV_TIMEOUT
RESTORE
RETRIES
RETRY_COUNT
RETRY_DELAY
SDU
SECURITY
SEND_BUF_SIZE
SERVER
SERVICE_NAME

Parameter
SERVICE_TAG
SESSION_POOL
SHARDING_KEY
SID
SSL_SERVER_CERT_DN
SSL_SERVER_DN_MATCH
STATEMENT_CACHE_SIZE
SUPER_SHARDING_KEY
TRANSPORT_CONNECT_TIMEOUT
TYPE_OF_SERVICE
WALLET_LOCATION

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.2.2 Naming-Restricted Parameter Values

For some of the naming-restricted parameter names, some values are not allowed on the right-hand side (after =).

Note these restrictions on the values that you can specify:

Parameter	Restriction
PROTOCOL	BEQ or Fork capability is not allowed. The allowed values are: • TCP • TCPS • WSS • EXADIRECT
WALLET_LOCATION	Only SYSTEM is allowed. For the METHOD sub-parameter, the allowed values are: • OCIVAULT • AZUREVAULT • BASIC • PRECONNECT

> **Note:**
>
> To disable these naming restrictions, set the sqlnet.ora parameter or environment variable NAMES.CLOUD_NAMING_RESTRICTIONS to FALSE. The default value for this parameter is TRUE.

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.3 Authentication Parameters for Azure App Configuration Store

These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Azure App Configuration store.

- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.
- AZURE_CLIENT_ID
  Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.
- AZURE_TENANT_ID
  Use the `AZURE_TENANT_ID` parameter to specify the tenant ID associated with the Microsoft Azure Active Directory (Azure AD) subscription.
- AZURE_CLIENT_SECRET
  Use the `AZURE_CLIENT_SECRET` parameter to specify a client secret for your Microsoft Azure Active Directory (Azure AD) application.
- AZURE_CLIENT_CERTIFICATE_PATH
  Use the `AZURE_CLIENT_CERTIFICATE_PATH` parameter to specify a client certificate of the Microsoft Azure Active Directory (Azure AD) application.
- AZURE_MANAGED_IDENTITY_CLIENT_ID
  Use the `AZURE_MANAGED_IDENTITY_CLIENT_ID` parameter to specify a user-assigned managed identity for authentication with Microsoft Azure Active Directory (Azure AD).
- HTTPS_PROXY
  Use the `HTTPS_PROXY` parameter to specify the HTTPS proxy host name when the client is behind a corporate HTTPS proxy.
- HTTPS_PROXY_PORT
  Use the `HTTPS_PROXY_PORT` parameter to specify the HTTPS proxy port when the client is behind a corporate HTTPS proxy.
- TIMEOUT
  Use the `TIMEOUT` parameter to specify the duration to complete each HTTP call.

7.3.1 AUTHENTICATION

Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

Purpose

To control how the Centralized Configuration Provider component authenticates with the Azure App Configuration store.

This is an optional parameter. If you do not set authentication method, then the default setting (AUTHENTICATION=AZURE_DEFAULT) is used to provide the Azure Service Principal flow. Set this parameter only if you want to override this default flow.

Usage Notes

Specify this parameter in the connect identifier string at the command line using the option=value syntax. Depending on the specified authentication method, you must additionally set the corresponding authentication parameters in the same string as given below:

- AZURE_DEFAULT provides the default token credential authentication using the Azure Service Principal (or OAuth 2.0 Client Credential) flow.

  This authenticates to Azure Active Directory (Azure AD) as a service principal using either a client secret or an X509 client certificate assigned during App Registration.

  The client driver evaluates the following credential types (if enabled) in an order:

  1. Azure Service Principal with Client Secret Credentials:
     a. The driver checks if the AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_SECRET parameters are set at the command line.
     b. If the preceding parameters are not set, then it tries the AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_SECRET environment variables. If configured, then the driver authenticates as a service principal using client secret. Otherwise, the driver proceeds to the next step.
  2. Azure Service Principal with Client Certificate Credentials:
     a. The driver checks if the AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_CERTIFICATE_PATH parameters are set at the command line.
     b. If the preceding parameters are not set, then it tries the AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_CERTIFICATE_PATH environment variables. If configured, then the driver authenticates as a service principal using client certificate.

  > **Note:**
  >
  > For both of these flows, if all the environment variables are set (that is, AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_SECRET, and AZURE_CLIENT_CERTIFICATE_PATH), then the driver first attempts AZURE_TENANT_ID, AZURE_CLIENT_ID and AZURE_SECRET_ID values.

- AZURE_SERVICE_PRINCIPAL provides the Azure Service Principal flow.

  Similar to the default flow, this authenticates the request as a service principal using either the client secret or client certificate assigned during App Registration. However, here the driver looks only at the command line parameters, as follows:

  1. The driver first tries the AZURE_TENANT_ID, AZURE_CLIENT_ID, and AZURE_CLIENT_SECRET parameters set at the command line.

2. If client secret is not set, then the driver looks for client certificate by trying the `AZURE_TENANT_ID`, `AZURE_CLIENT_ID`, and `AZURE_CLIENT_CERTIFICATE_PATH` parameters.

> **Note:**
>
> If both the `AZURE_CLIENT_SECRET` and `AZURE_CLIENT_CERTIFICATE_PATH` parameters are set, then this flow uses client certificate for authentication.

- `AZURE_MANAGED_IDENTITY` provides either the Azure Managed Identity flow or the Azure Managed User Identity flow.

  This method works for Azure Virtual Machines, App Service, Azure Functions applications, and the Azure Cloud Shell. This attempts authentication to Azure AD using a managed identity assigned to the deployment environment.

  If you want to use a "user-assigned" managed identity for authentication, then you must additionally specify the `AZURE_MANAGED_IDENTITY_CLIENT_ID` authentication parameter.

Values

- `AZURE_DEFAULT` to enable the default token credential authentication flow
- `AZURE_SERVICE_PRINCIPAL` to enable authentication using client credentials set at the command line
- `AZURE_MANAGED_IDENTITY` to enable authentication using managed identity or managed user identity credentials

Default

```
AZURE_DEFAULT
```

Example

```
AUTHENTICATION=AZURE_SERVICE_PRINCIPAL
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- Authentication Parameters for Azure App Configuration Store
  These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Azure App Configuration store.

7.3.2 AZURE_CLIENT_ID

Use the `CLIENT_ID` parameter to specify the ID of the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify the client ID associated with Azure App Registration for authenticating to Azure AD.

You use this parameter for the AZURE_DEFAULT and AZURE_SERVICE_PRINCIPAL authentication flows to access the Azure App Configuration store.

Usage Notes

Set this parameter at the command line in the connect identifier using the option=value syntax.

Default

None

Value

Client (application) ID of an App Registration in the tenant. You can get this value from the registered application essentials page in the Azure portal.

Example

```
AZURE_CLIENT_ID=123ab4cd-1a2b-1234-a12b-aa00123b2cd3
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the AUTHENTICATION parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

7.3.3 AZURE_TENANT_ID

Use the AZURE_TENANT_ID parameter to specify the tenant ID associated with the Microsoft Azure Active Directory (Azure AD) subscription.

Purpose

To specify the tenant ID associated with the Azure AD subscription. Azure AD is trusted to authenticate users, services, and devices for the subscription.

You use this parameter for the AZURE_DEFAULT and AZURE_SERVICE_PRINCIPAL authentication flows to access the Azure App Configuration store.

Usage Notes

Set this parameter at the command line in the connect identifier using the option=value syntax.

Default

None

Value

Azure AD tenant ID associated with Azure subscription. You can get this value from the Overview page under Basic information in the Azure portal.

Example

```
AZURE_TENANT_ID=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

7.3.4 AZURE_CLIENT_SECRET

Use the `AZURE_CLIENT_SECRET` parameter to specify a client secret for your Microsoft Azure Active Directory (Azure AD) application.

Purpose

To specify a client secret associated with Azure App Registration for authenticating to Azure AD. This credential enables confidential applications to identify themselves to the authentication service, when receiving tokens at a web address location (using an HTTPS scheme).

You use this parameter for the `AZURE_DEFAULT` and `AZURE_SERVICE_PRINCIPAL` authentication flows to access the Azure App Configuration store.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

Client secret generated during App Registration. This value appears on the Certificates and Secrets page when you create a new client secret in the Azure portal.

Example

```
AZURE_CLIENT_SECRET=a1abcd12abcd1abc1abc1abc12a
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

7.3.5 AZURE_CLIENT_CERTIFICATE_PATH

Use the `AZURE_CLIENT_CERTIFICATE_PATH` parameter to specify a client certificate of the Microsoft Azure Active Directory (Azure AD) application.

Purpose

To enable authentication to Azure AD as a service principal, using an X509 certificate assigned during App Registration.

You use this parameter for the `AZURE_DEFAULT` and `AZURE_SERVICE_PRINCIPAL` authentication flows to access the Azure App Configuration store.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

File system path to a `PEM` or `PKCS12` certificate file including the private key. This certificate file must have a private key without password. Ensure that you protect this certificate with necessary file permissions.

Example

```
AZURE_CLIENT_CERTIFICATE_PATH=ORACLE_HOME/.azure/certificates/my-
app.pem
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

7.3.6 AZURE_MANAGED_IDENTITY_CLIENT_ID

Use the `AZURE_MANAGED_IDENTITY_CLIENT_ID` parameter to specify a user-assigned managed identity for authentication with Microsoft Azure Active Directory (Azure AD).

Purpose

To specify a user-assigned managed identity if you are using the Managed User Identity authentication flow for accessing the Azure App Configuration store. This managed identity acts as a standalone Azure AD resource.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax, along with the `AUTHENTICATION=AZURE_MANAGED_IDENTITY` setting.

For example:

```
sqlplus /@"config-azure://dbclient-appconfig?key=/database/hr/
&authentication=azure_managed_identity&azure_managed_identity_client_id=b1c12
34-1a12-1234-ab12-3a1a1bc12a"
```

Default

None

Value

Client ID for a user-assigned managed identity. You can get this value from the Managed Identities - Overview page in the Azure portal.

Example

```
AZURE_MANAGED_IDENTITY_CLIENT_ID=b1c1234-1a12-1234-ab12-3a1a1bc12a
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- Managed Identities for Azure Resources Documentation
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Azure App Configuration store.

7.3.7 HTTPS_PROXY

Use the `HTTPS_PROXY` parameter to specify the HTTPS proxy host name when the client is behind a corporate HTTPS proxy.

Purpose

To specify the HTTPS proxy host name for tunneling Transport Layer Security (TLS) client connections, when the client is behind a corporate HTTPS proxy.

Default

None

Value

HTTPS proxy host name

Examples

```
HTTPS_PROXY=www-proxy.example.com
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.3.8 HTTPS_PROXY_PORT

Use the `HTTPS_PROXY_PORT` parameter to specify the HTTPS proxy port when the client is behind a corporate HTTPS proxy.

Purpose

To specify the HTTPS proxy port for tunneling TLS client connections, when the client is behind a corporate HTTPS proxy.

Default

None

Value

HTTPS proxy port number

Example

```
HTTPS_PROXY_PORT=80
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.3.9 TIMEOUT

Use the `TIMEOUT` parameter to specify the duration to complete each HTTP call.

Purpose

To specify the duration to complete each HTTP call. Set this parameter to avoid long network delays or indefinite waiting for a server's response.

Value

Timeout on each HTTP call, in seconds

Default

None

Example

```
TIMEOUT=60
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.4 Authentication Parameters for OCI Object Storage

These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.
- OCI_PROFILE
  Use the `OCI_PROFILE` parameter to specify the profile file section name.
- OCI_PROFILE_PATH
  Use the `OCI_PROFILE_PATH` parameter to override the default Oracle Cloud Infrastructure (OCI) configuration file location.
- OCI_TENANCY
  Use the `OCI_TENANCY` parameter to specify the tenant ID associated with Oracle Cloud Infrastructure (OCI) subscription.
- OCI_USER
  Use the `OCI_USER` parameter to specify the user ID associated with Oracle Cloud Infrastructure (OCI) subscription.
- OCI_KEY_FILE
  Use the `OCI_KEY_FILE` parameter to specify location of the Oracle Cloud Infrastructure (OCI) private key file.
- OCI_FINGERPRINT
  Use the `OCI_FINGERPRINT` parameter to specify the fingerprint of the Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) profile key.
- HTTPS_PROXY
  Use the `HTTPS_PROXY` parameter to specify the HTTPS proxy host name when the client is behind a corporate HTTPS proxy.
- HTTPS_PROXY_PORT
  Use the `HTTPS_PROXY_PORT` parameter to specify the HTTPS proxy port when the client is behind a corporate HTTPS proxy.
- TIMEOUT
  Use the `TIMEOUT` parameter to specify the duration to complete each HTTP call.

7.4.1 AUTHENTICATION

Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

Purpose

To control how the Centralized Configuration Provider component authenticates with the OCI Object Storage JSON file.

This is an optional parameter. If you do not set authentication method, then the default setting (`AUTHENTICATION=OCI_DEFAULT`) is used to provide the OCI API Key flow. Set this parameter only if you want to override this default flow.

Usage Notes

Specify this parameter in the connect identifier using the `option=value` syntax. Depending on the specified authentication method, you must additionally set the corresponding authentication parameters in the same string as given below:

- `OCI_DEFAULT` provides the default profile-based OCI authentication using the OCI API Key flow. This flow enables authentication to OCI using API key-related values.

  The client driver evaluates the following credential types (if enabled) in an order:

  1. It first checks if the `OCI_TENANCY`, `OCI_USER`, `OCI_FINGERPRINT`, and `OCI_KEY_FILE` parameters are set at the command line.
  2. If the preceding parameters are not set, then it checks for these parameters in the OCI configuration file present at the default location (`~/.oci/config`) or at the location specified by the `OCI_CONFIG_FILE` environment variable.

     The driver then checks if the file is present at the location configured by the `OCI_PROFILE_PATH` parameter.

     Finally, the driver checks if the file contains a profile matching the name configured by the `OCI_PROFILE` parameter or the default name (`DEFAULT`).
- `OCI_INSTANCE_PRINCIPAL` provides the OCI Instance Principal flow. This flow enables authentication to OCI using VM instance credentials running on OCI.
- `OCI_RESOURCE_PRINCIPAL` provides the OCI Resource Principal flow. This flow enables authentication to OCI using OCI resource principals.

Default

```
OCI_DEFAULT
```

Values

- `OCI_DEFAULT` to enable the default profile-based OCI authentication using API key-related values
- `OCI_INSTANCE_PRINCIPAL` to enable authentication using VM instance credentials
- `OCI_RESOURCE_PRINCIPAL` to enable authentication using OCI resource principals

Example

```
AUTHENTICATION=OCI_RESOURCE_PRINCIPAL
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- OCI SDK Authentication Methods
- Authentication Parameters for OCI Object Storage
  These are the authentication parameters that you specify in the connect identifier (using the `option=value` syntax) to access the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.2 OCI_PROFILE

Use the `OCI_PROFILE` parameter to specify the profile file section name.

Purpose

To specify the profile file section name. This profile is the client connection information stored in your Oracle Cloud Infrastructure (OCI) configuration file.

You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

`DEFAULT`

Value

Profile file section name to use for authentication to OCI.

Example

```
OCI_PROFILE=ADMIN_USER
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.3 OCI_PROFILE_PATH

Use the `OCI_PROFILE_PATH` parameter to override the default Oracle Cloud Infrastructure (OCI) configuration file location.

Purpose

To override the default OCI configuration file location. You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

If you do not set this parameter, then the client driver looks for the OCI configuration file either at the default location (`~/.oci/config`) or at the location specified by the `OCI_CONFIG_FILE` environment variable.

Default

None

Value

Full path (including a file name) to the OCI configuration file

Example

```
OCI_PROFILE_PATH=/app/myociprofile/config
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.4 OCI_TENANCY

Use the `OCI_TENANCY` parameter to specify the tenant ID associated with Oracle Cloud Infrastructure (OCI) subscription.

Purpose

To specify the tenant ID associated with OCI subscription. OCI ID is trusted to authenticate users, services, and devices for the subscription.

You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

Tenancy's Oracle Cloud Identifier (OCID). You can get this value from the Tenancy Details page under Tenancy Information in the OCI console.

Example

```
OCI_TENANCY=1a123ab1-a1b1-1a2b-a1b2-a12bcdab0123
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.5 OCI_USER

Use the `OCI_USER` parameter to specify the user ID associated with Oracle Cloud Infrastructure (OCI) subscription.

Purpose

To specify the user ID associated with OCI subscription. You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

User's Oracle Cloud Identifier (OCID). You can get this value from the User Details page under User Information in the OCI console.

Example

```
OCI_USER=ocid1.user.oc1..aa1abcd12bc1abcd1abcdf1abcdfg12abcd
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.6 OCI_KEY_FILE

Use the `OCI_KEY_FILE` parameter to specify location of the Oracle Cloud Infrastructure (OCI) private key file.

Purpose

To specify location of the OCI private key file. You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

If you want to make API requests, then you must obtain an RSA public key in PEM format (minimum 2048 bits) added to your OCI Identity Access Management (IAM) user profile and sign the API requests with the corresponding private key.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

OCI private key file location. This value must be corresponding to the public key added to the IAM user profile.

Example

```
OCI_KEY_FILE=~/.oci/oci_api_key.pem
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.7 OCI_FINGERPRINT

Use the `OCI_FINGERPRINT` parameter to specify the fingerprint of the Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) profile key.

Purpose

To specify the fingerprint of the key added to the OCI IAM profile. You use this parameter for the OCI API Key authentication flow to access the OCI Object Storage JSON file.

Usage Notes

Set this parameter at the command line in the connect identifier using the `option=value` syntax.

Default

None

Value

Key's fingerprint. You can get this value from the User Details page under API Keys in the OCI console.

Example

```
OCI_FINGERPRINT=a1:12:a1:1a:01:12:23:34:ab:12:ab:0a:12:a1:
```

Related Topics

- Oracle Cloud Infrastructure Documentation
- *Oracle Database Net Services Administrator's Guide*
- AUTHENTICATION
  Use the `AUTHENTICATION` parameter to specify the authentication method that you want to use for accessing the Oracle Cloud Infrastructure (OCI) Object Storage JSON file.

7.4.8 HTTPS_PROXY

Use the `HTTPS_PROXY` parameter to specify the HTTPS proxy host name when the client is behind a corporate HTTPS proxy.

Purpose

To specify the HTTPS proxy host name for tunneling Transport Layer Security (TLS) client connections, when the client is behind a corporate HTTPS proxy.

Default

None

Value

HTTPS proxy host name

Examples

```
HTTPS_PROXY=www-proxy.example.com
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.4.9 HTTPS_PROXY_PORT

Use the `HTTPS_PROXY_PORT` parameter to specify the HTTPS proxy port when the client is behind a corporate HTTPS proxy.

Purpose

To specify the HTTPS proxy port for tunneling TLS client connections, when the client is behind a corporate HTTPS proxy.

Default

None

Value

HTTPS proxy port number

Example

```
HTTPS_PROXY_PORT=80
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

7.4.10 TIMEOUT

Use the `TIMEOUT` parameter to specify the duration to complete each HTTP call.

Purpose

To specify the duration to complete each HTTP call. Set this parameter to avoid long network delays or indefinite waiting for a server's response.

Value

Timeout on each HTTP call, in seconds

Default

None

Example

```
TIMEOUT=60
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

8

Oracle Net Listener Parameters in the listener.ora File

This chapter provides a complete listing of the `listener.ora` file configuration parameters.

- Overview of Oracle Net Listener Configuration File
  Oracle Net Listener configuration, stored in the `listener.ora` file, consists of these elements.
- Protocol Address Parameters
- Connection Rate Limiter Parameters
  The connection rate limiter feature in Oracle Net Listener enables a database administrator to limit the number of new connections handled by the listener. When this feature is enabled, Oracle Net Listener imposes a user-specified maximum limit on the number of new connections handled by the listener every second. Depending on the configuration, the rate can be applied to a collection of endpoints, or to a specific endpoint.
- Control Parameters
  This section describes the following parameters that control the behavior of the listener:
- ADR Diagnostic Parameters for Oracle Net Listener
  The diagnostic data for the critical errors is quickly captured and stored in the ADR for Oracle Net listener.
- Non-ADR Diagnostic Parameters for Oracle Net Listener
  This section lists the parameters used when ADR is disabled. The default value of `DIAG_ADR_ENABLED_`*`listener_name`* is `on`. Therefore, the `DIAG_ADR_ENABLED_`*`listener_name`* parameter *must* explicitly be set to `off` to use non-ADR tracing.
- Class of Secure Transports Parameters
  The class of secure transports (COST) parameters specify a list of transports that are considered secure for administration and registration of a particular listener.

8.1 Overview of Oracle Net Listener Configuration File

Oracle Net Listener configuration, stored in the `listener.ora` file, consists of these elements.

- Name of the listener
- Protocol addresses that the listener is accepting connection requests on
- Valid nodes that the listener allows to register with the database
- Database services
- Control parameters

Dynamic service registration, eliminates the need for static configuration of supported services. However, static service configuration is required if you plan to use Oracle Enterprise

Manager Cloud Control. For information about static service configuration, see *Oracle Database Net Services Administrator's Guide*.

By default, the `listener.ora` file is located in the *`ORACLE_HOME`*`/network/admin` directory. You can also store the `listener.ora` in the following locations:

- The directory specified by the `TNS_ADMIN` environment variable or registry value.
- On Linux and UNIX operating systems, it is the global configuration directory. For example, on the Oracle Solaris operating system, the directory is `/var/opt/oracle`. See *Oracle Database Global Data Services Concepts and Administration Guide* for information about management of global service. Also refer to Oracle operating system-specific documentation.
- In the read-only Oracle home mode, the default location for the `listener.ora` file is *`ORACLE_BASE_HOME`*`/network/admin`. If the `listener.ora` file is not present in the *`ORACLE_BASE_HOME`*`/network/admin` directory, then search for the file in the *`ORACLE_HOME`*`/network/admin` directory.
- In the read-only Oracle home mode, the parameters are stored in the `ORACLE_BASE_HOME` location by default.

It is possible to configure multiple listeners, each with a unique name, in one `listener.ora` file. Multiple listener configurations are possible because each of the top-level configuration parameters has a suffix of the listener name or is the listener name itself.

> **Note:**
>
> - It is often useful to configure multiple listeners in one `listener.ora` file. However, Oracle recommends running only one listener for each node in most customer environments.
> - Oracle Net Services supports the IFILE parameter in the `listener.ora` file, with up to three levels of nesting. The parameter is added manually to the file. The following is an example of the syntax:
>
>   ```
>   IFILE=/tmp/listener_em.ora
>   IFILE=/tmp/listener_cust1.ora
>   IFILE=/tmp/listener_cust2.ora
>   ```
>
>   Refer to *Oracle Database Reference* for additional information.

The following example shows a `listener.ora` file for a listener named `LISTENER`, which is the default name of the listener.

Example 8-1 listener.ora File

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sale-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))))
```

8.2 Protocol Address Parameters

The protocol address section of the `listener.ora` file defines the protocol addresses on which the listener is accepting connection requests. This section describes the most common parameters used in protocol addresses. The `ADDRESS_LIST` parameter is also supported. This section lists and describes the following parameters:

- ADDRESS
  The protocol `ADDRESS` parameter's networking parameter is in the `listener.ora` file. It specifies the protocol address under the `DESCRIPTION` parameter for one listener.
- DESCRIPTION
  `DESCRIPTION` networking parameter of the listener.ora file contains listener protocol addresses.
- Firewall
- IP
  The protocol address parameter `IP` determine which IP address the listener listens on when a host name is specified
- QUEUESIZE
- RECV_BUF_SIZE
  Use the `RECV_BUF_SIZE` parameter to specify buffer space for session receive operations.
- SEND_BUF_SIZE
  Use the `SEND_BUF_SIZE` parameter to specify buffer space for session send operations.

8.2.1 ADDRESS

The protocol `ADDRESS` parameter's networking parameter is in the `listener.ora` file. It specifies the protocol address under the `DESCRIPTION` parameter for one listener.

Purpose

Specifies a single listener protocol address in the `DESCRIPTION` parameter

Usage Notes

Use this parameter to define the protocol, the host, and the port number for the listener.

Example

```
listener_name=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=hr-server)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))))
```

8.2.2 DESCRIPTION

`DESCRIPTION` networking parameter of the listener.ora file contains listener protocol addresses.

Purpose

To contain listener protocol addresses.

Example 8-2 Example

listener_name

8.2.3 Firewall

Purpose

It can be set in endpoint to enable firewall functionality.

Related Topics

- *Oracle Database Net Services Administrator's Guide*

8.2.4 IP

The protocol address parameter `IP` determine which IP address the listener listens on when a host name is specified

Purpose

To determine which IP address the listener listens on when a host name is specified.

Usage Notes

This parameter is only applicable when the `HOST` parameter specifies a host name.

Values

- `first`

  Listen on the first IP address returned by the DNS resolution of the host name. If the user wants the listener to listen on the first IP to which the specified host name resolves, then the address must be qualified with `(IP=first)`.

- `v4_only`

  Listen only on IPv4 addresses.

- `v6_only`

  Listen only on IPv6 addresses.

Default

This feature is disabled by default.

Example

```
listener_name=
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=rancode1-vip)(PORT=1522)(IP=v6_only))
```

8.2.5 QUEUESIZE

Purpose

To specify the number of concurrent connection requests that the listener can accept on a TCP/IP or IPC listening endpoint (protocol address).

Usage Notes

The number of concurrent connection requests is dependent on the platform and listener usage scenarios. If the listener is heavily-loaded, then set the parameter to a higher number.

Put this parameter at the end of the protocol address with its value set to the expected number of concurrent connection requests.

Default

The default number of concurrent connection requests is operating system specific.

Example

```
listener_name=
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=hr-server)(PORT=1521)(QUEUESIZE=20)))
```

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for additional information about configuring this parameter

8.2.6 RECV_BUF_SIZE

Use the `RECV_BUF_SIZE` parameter to specify buffer space for session receive operations.

Purpose

To specify, in bytes, the buffer space for receive operations of sessions.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter or at the end of the protocol address with its value set to the expected number of bytes.

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to the operating system-specific documentation for information about additional protocols that support this parameter.

Default

The default value for this parameter is operating system specific. The default for the Linux operating system is 87380 bytes.

Example

```
listener_name=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
        (RECV_BUF_SIZE=11784))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc)
        (RECV_BUF_SIZE=11784))))
listener_name=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (RECV_BUF_SIZE=11784))
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

8.2.7 SEND_BUF_SIZE

Use the `SEND_BUF_SIZE` parameter to specify buffer space for session send operations.

Purpose

To specify, in bytes, the buffer space for send operations of sessions.

Usage Notes

Put this parameter under the `DESCRIPTION` parameter or at the end of the protocol address.

This parameter is supported by the TCP/IP, TCP/IP with TLS, and SDP protocols.

> **Note:**
>
> Additional protocols might support this parameter on certain operating systems. Refer to operating system-specific documentation for additional information about additional protocols that support this parameter.

Default

The default value for this parameter is operating system specific. The default for the Linux operating system is 16 KB.

Example

```
listener_name=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
       (SEND_BUF_SIZE=11280))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc)
       (SEND_BUF_SIZE=11280))))
listener_name=
  (DESCRIPTION=
    (SEND_BUF_SIZE=11280)
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*

8.3 Connection Rate Limiter Parameters

The connection rate limiter feature in Oracle Net Listener enables a database administrator to limit the number of new connections handled by the listener. When this feature is enabled, Oracle Net Listener imposes a user-specified maximum limit on the number of new connections handled by the listener every second. Depending on the configuration, the rate can be applied to a collection of endpoints, or to a specific endpoint.

This feature is controlled through the following `listener.ora` configuration parameters:

- CONNECTION_RATE_listener_name
  The `CONNECTION_RATE_listener_name` configuration parameter of the`listener.ora` file specifies a global rate that is enforced across all listening endpoints that are rate-limited.
- RATE_LIMIT
  The `RATE_LIMIT` configuration parameter of the `listener.ora` file indicates that a particular listening endpoint is rate-limited.

8.3.1 CONNECTION_RATE_listener_name

The `CONNECTION_RATE_listener_name` configuration parameter of the`listener.ora` file specifies a global rate that is enforced across all listening endpoints that are rate-limited.

Purpose

To specify a global rate that is enforced across all listening endpoints that are rate-limited.

Usage Notes

When this parameter is specified, it overrides any endpoint-level numeric rate values that might be specified.

Syntax

```
CONNECTION_RATE_listener_name=number_of_connections_per_second
```

8.3.2 RATE_LIMIT

The `RATE_LIMIT` configuration parameter of the `listener.ora` file indicates that a particular listening endpoint is rate-limited.

Purpose

To indicate that a particular listening endpoint is rate-limited.

Usage Notes

The parameter is specified in the `ADDRESS` section of the listener endpoint configuration.

Syntax

```
LISTENER=
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521)(RATE_LIMIT=yes))
```

- When the `RATE_LIMIT` parameter is set to `yes` for an endpoint, that endpoint is included in the enforcement of the global rate configured by the `CONNECTION_RATE_listener_name` parameter. The global rate limit is enforced individually at each endpoint that has `RATE_LIMIT` set to `yes`.
- Dynamic endpoints for listeners managed by Oracle Clusterware have the `RATE_LIMIT` parameter set to `yes`.
- When the `RATE_LIMIT` parameter is set to a value greater than `0`, then the rate limit is enforced at that endpoint level.

Examples

The following examples use the `CONNECTION_RATE_listener name` and `RATE_LIMIT` parameters.

Example 1

```
CONNECTION_RATE_LISTENER=10

LISTENER=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521)(RATE_LIMIT=yes))
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1522)(RATE_LIMIT=yes))
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1523)))
```

In the preceding example, the global rate of new connections is enforced separately for each endpoint. Connections through port 1521 are limited at 10 every second, and the connections through port 1522 are also separately limited at 10 every second. Connections through port 1523 are not limited.

Example 2

```
LISTENER= (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521)(RATE_LIMIT=5))
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1522)(RATE_LIMIT=10))
   (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1523))
   )
```

In the preceding example, the connection rates are enforced at the endpoint level. A maximum of 5 connections are processed through port 1521 every second. The limit for connections through port 1522 is 10 every second. Connections through port 1523 are not limited.

> **Note:**
>
> The global `CONNECTON_RATE_listener_name` parameter is not specified in the preceding configuration. If it is specified, then the limits on ports 1521 and 1522 are ignored, and the global value is used instead.

8.4 Control Parameters

This section describes the following parameters that control the behavior of the listener:

- ADMIN_RESTRICTIONS_listener_name
  The `listener.ora` control parameter `ADMIN_RESTRICTIONS_listener_name` restricts runtime administration of the listener.
- ALLOW_MULTIPLE_REDIRECTS_listener_name
  The *listener.ora* control parameter `ALLOW_MULTIPLE_REDIRECTS_listener_name` enables multiple redirects of the client.
- CRS_NOTIFICATION_listener_name
  `CRS_NOTIFICATION_listener_name` control parameter of the `listener.ora` file sets notification to allow or disallow Cluster Ready Services (CRS) to manage the listener in an Oracle Real Application Clusters environment.
- DEDICATED_THROUGH_BROKER_LISTENER
  `DEDICATED_THROUGH_BROKER_LISTENER` networking parameter of the `listener.ora` file enables the server to spawn a thread or process when a connection to the database is requested through the listener.
- DEFAULT_SERVICE_listener_name
  `DEFAULT_SERVICE_listener_name` control parameter of the `listener.ora` file enables users to connect to the database without having to specify a service name from the client side.
- ENABLE_EXADIRECT_listener_name
- INBOUND_CONNECT_TIMEOUT_listener_name
- LOCAL_REGISTRATION_ADDRESS_listener_name

- MAX_ALL_CONNECTIONS_listener_name
  Use the `MAX_ALL_CONNECTIONS_listener_name` parameter to specify the maximum number of concurrent registration and client connection sessions.
- MAX_REG_CONNECTIONS_listener_name
  Use the `MAX_REG_CONNECTIONS_listener_name` parameter to specify the maximum number of concurrent registration connection sessions.
- REGISTRATION_EXCLUDED_NODES_listener_name
- REGISTRATION_INVITED_NODES_listener_name
- REMOTE_REGISTRATION_ADDRESS_listener_name
- SAVE_CONFIG_ON_STOP_listener_name
- SERVICE_RATE_listener_name
  The `SERVICE_RATE_listener_name` control parameter specifies incoming connection rate that is allowed per service for an instance.
- SSL_CIPHER_SUITES
  Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).
- SSL_CLIENT_AUTHENTICATION
  Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).
- SSL_VERSION
  Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.
- SUBSCRIBE_FOR_NODE_DOWN_EVENT_listener_name
- USE_SID_AS_SERVICE_listener_name
- VALID_NODE_CHECKING_REGISTRATION_listener_name
  The *listener.ora* control parameter `VALID_NODE_CHECKING_REGISTRATION_listener_name` determines if valid node checking registration is performed, or if the subnet is allowed.
- WALLET_LOCATION
  Use the `WALLET_LOCATION` parameter to specify the location of Oracle wallets.

8.4.1 ADMIN_RESTRICTIONS_*listener_name*

The `listener.ora` control parameter `ADMIN_RESTRICTIONS_listener_name` restricts runtime administration of the listener.

Purpose

To restrict runtime administration of the listener.

Usage Notes

Setting `ADMIN_RESTRICTIONS_listener_name=on` disables the runtime modification of parameters in `listener.ora`. That is, the listener refuses to accept SET commands that alter its parameters. To change any of the parameters in `listener.ora`, including `ADMIN_RESTRICTIONS_listener_name` itself, modify the `listener.ora` file manually and

reload its parameters using the RELOAD command for the new changes to take effect without explicitly stopping and restarting the listener.

Default

`off`

Example

```
ADMIN_RESTRICTIONS_listener=on
```

Related Topics

- SET
  Use the Listener Control utility command `SET` to alter listener parameter values.
- RELOAD
  Use the Listener Control utility command `RELOAD` to reload the `listener.ora` file so that you can add or change statically configured services without stopping the listener.

8.4.2 ALLOW_MULTIPLE_REDIRECTS_*listener_name*

The *listener.ora* control parameter `ALLOW_MULTIPLE_REDIRECTS_listener_name` enables multiple redirects of the client.

Purpose

To support multiple redirects of the client.

Usage Notes

This parameter should only be set on the SCAN listener on the Oracle Public Cloud. When set to `on`, multiple redirects of the client are allowed.

Do not set this parameter for a node listener if that is used as a SCAN listener.

Default

off

Values

`on | off`

Example

```
ALLOW_MULTIPLE_REDIRECTS_listener=on
```

8.4.3 CRS_NOTIFICATION_listener_name

`CRS_NOTIFICATION_listener_name` control parameter of the `listener.ora` file sets notification to allow or disallow Cluster Ready Services (CRS) to manage the listener in an Oracle Real Application Clusters environment.

Purpose

To set notification.

Usage Notes

By default, the Oracle Net listener notifies Cluster Ready Services (CRS) when it is started or stopped. These notifications allow CRS to manage the listener in an Oracle Real Application Clusters environment. This behavior can be prevented by setting the `CRS_NOTIFICATION_listener_name` parameter to `off`.

Default

`on`

Values

`on | off`

8.4.4 DEDICATED_THROUGH_BROKER_LISTENER

`DEDICATED_THROUGH_BROKER_LISTENER` networking parameter of the `listener.ora` file enables the server to spawn a thread or process when a connection to the database is requested through the listener.

Purpose

To enable the server to spawn a thread or process when a connection to the database is requested through the listener.

Default

`off`

Values

`on | off`

Example 8-3 Example

(Optional) Enter an example to illustrate your reference here.

8.4.5 DEFAULT_SERVICE_listener_name

`DEFAULT_SERVICE_listener_name` control parameter of the `listener.ora` file enables users to connect to the database without having to specify a service name from the client side.

Purpose

To enable users to connect to the database without having to specify a service name from the client side.

Usage Notes

When a client tries to connect to the database, the connection request passes through the listener. The listener may be servicing several different databases. If a service name is configured in this parameter, then users may not necessarily need to specify a service name in the connect syntax. If a user specifies a service name, then the listener connects the user to that specific database, otherwise the listener connects to

the service name specified by the DEFAULT_SERVICE_*listener_name* parameter. For container databases, the client must explicitly specify the service name.

Default

There is no default value for the DEFAULT_SERVICE_*listener_name* parameter. If this parameter is not configured and a user does not specify a fully-qualified service name in the connect syntax, then the connection attempt fails. This parameter only accepts one value.

Example 8-4 Example

```
DEFAULT_SERVICE_listener=sales.us.example.com
```

8.4.6 ENABLE_EXADIRECT_*listener_name*

Purpose

To enable Exadirect protocol.

Usage Notes

The parameter enables Exadirect support.

Default

```
Off
```

Values

```
on | off
```

Example 8-5 Example

```
ENABLE_EXADIRECT_listener=on
```

8.4.7 INBOUND_CONNECT_TIMEOUT_*listener_name*

Purpose

To specify the time, in seconds, for the client to complete its connect request to the listener after the network connection had been established.

Usage Notes

If the listener does not receive the client request in the time specified, then it terminates the connection. In addition, the listener logs the IP address of the client and an `ORA-12525:TNS: listener has not received client's request in time allowed` error message to the `listener.log` file.

To protect both the listener and the database server, Oracle recommends setting this parameter in combination with the SQLNET.INBOUND_CONNECT_TIMEOUT parameter in the `sqlnet.ora` file. When specifying values for these parameters, consider the following recommendations:

- Set both parameters to an initial low value.
- Set the value of the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter to a lower value than the SQLNET.INBOUND_CONNECT_TIMEOUT parameter.

For example, you can set the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter to 2 seconds and the INBOUND_CONNECT_TIMEOUT parameter to 3 seconds. If clients are unable to complete connections within the specified time due to system or network delays that are normal for the particular environment, then increment the time as needed.

Default

60 seconds

Example

```
INBOUND_CONNECT_TIMEOUT_listener=2
```

8.4.8 LOCAL_REGISTRATION_ADDRESS_listener_name

Purpose

To secure registration requests through dedicated secure registration endpoints for local listeners. Service ACLs are accepted by listener only if LOCAL_REGISTRATION_ADDRESS_*lsnr alias* is configured. The parameter specifies the group that is allowed to send ACLs.

Usage Notes

The local registration endpoint accepts local registration connections from the specified group. All local registration requests coming on normal listening endpoints are redirected to the local registration endpoint. If the registrar is not a part of the group, then it cannot connect to the endpoint.

Default

```
OFF
```

Values

```
ON, OFF, or IPC endpoint address with group
```

When set to ON, listener defaults the group to oinstall on UNIX and ORA_INSTALL on Windows.

Example 8-6 Example

```
LOCAL_REGISTRATION_ADDRESS_lsnr_alias = (address=(protocol=ipc)
(group=xyz)) LOCAL_REGISTRATION_ADDRESS_lsnr_alias =ON
```

Related Topics

- Firewall
- DBMS_SFW_ACL_ADMIN

8.4.9 MAX_ALL_CONNECTIONS_*listener_name*

Use the `MAX_ALL_CONNECTIONS_listener_name` parameter to specify the maximum number of concurrent registration and client connection sessions.

Purpose

To specify the maximum number of concurrent registration and client connection sessions that can be supported by Oracle Net Listener.

Usage Notes

This number includes registration connections from databases, and ongoing client connection establishment requests. After a connection is established, the clients do not maintain a connection to the listener. This limit only applies to client connections that are in the initial connection establishment phase from a listener perspective.

Default

`4096`

Example

```
MAX_ALL_CONNECTIONS_listener=4096
```

8.4.10 MAX_REG_CONNECTIONS_*listener_name*

Use the `MAX_REG_CONNECTIONS_listener_name` parameter to specify the maximum number of concurrent registration connection sessions.

Purpose

To specify the maximum number of concurrent registration connection sessions that can be supported by Oracle Net Listener.

Default

`512`

Example

```
MAX_REG_CONNECTIONS_listener=2048
```

8.4.11 REGISTRATION_EXCLUDED_NODES_*listener_name*

Purpose

To specify the list of nodes that cannot register with the listener.

Usage Notes

The list can include host names or CIDR notation for IPv4 and IPv6 addresses. The wildcard format (`*`) is supported for IPv4 addresses. The presence of a host name in the list results in the inclusion of all IP addresses mapped to the host name. The host name should be consistent with the public network interface.

If the REGISTRATION_INVITED_NODES_*listener_name* parameter and the REGISTRATION_EXCLUDED_NODES_*listener_name* parameter are set, then the REGISTRATION_EXCLUDED_NODES_*listener_name* parameter is ignored.

Values

Valid nodes and subnet IP addresses or names.

Example

```
REGISTRATION_EXCLUDED_NODES_listener = (10.1.26.*, 10.16.40.0/24, \
                                        2001:DB8:3eff:fe38, node2)
```

8.4.12 REGISTRATION_INVITED_NODES_*listener_name*

Purpose

To specify the list of node that can register with the listener.

Usage Notes

- The list can include host names or CIDR notation for IPv4 and IPv6 addresses. The wildcard format (*) is supported for IPv4 addresses. The presence of a host name in the list results in the inclusion of all IP addresses mapped to the host name. The host name should be consistent with the public network interface.
- If the REGISTRATION_INVITED_NODES_*listener_name* parameter and the REGISTRATION_EXCLUDED_NODES_*listener_name* parameter are set, then the REGISTRATION_EXCLUDED_NODES_*listener_name* parameter is ignored.
- Starting with Oracle Grid Infrastructure 12c, for a SCAN listener, if the VALID_NODE_CHECKING_REGISTRATION_*listener_name* and REGISTRATION_INVITED_NODES_*listener_name* parameters are set in the listener.ora file, then the listener agent overwrites these parameters.

Values

Valid nodes and subnet IP addresses or names.

Example

```
REGISTRATION_INVITED_NODES_listener = (10.1.35.*, 10.1.34.0/24, \
                                       2001:DB8:fe38:7303, node1)
```

> **See Also:**
>
> *Oracle Real Application Clusters Administration and Deployment Guide* for information about valid node checking for registration

8.4.13 REMOTE_REGISTRATION_ADDRESS_*listener_name*

Purpose

To secure registration requests through dedicated secure registration endpoints for SCAN listeners.

Usage Notes

The registration endpoint is on a private network within the cluster. All remote registration requests coming in on normal listening endpoints are redirected to the registration endpoint. Any system which is not a part of the cluster cannot connect to the endpoint. This feature is not supported when `ADMIN_RESTRICTIONS_listener_name` is set to `ON` as the Cluster Ready Services agent configures the `remote_registration_address` dynamically at run time.

Default

This parameter is configured internally in listeners managed by Oracle Clusterware to restrict registrations to the private network. The value of this parameter should not be modified or specified explicitly. The only supported explicit setting is for turning this feature off by setting the value to `OFF`.

Values

```
off
```

Example

```
REMOTE_REGISTRATION_ADDRESS_listener=off
```

8.4.14 SAVE_CONFIG_ON_STOP_*listener_name*

Purpose

To specify whether runtime configuration changes are saved to the `listener.ora` file.

Usage Notes

When you set the parameter to `true`, any parameters that were modified while the listener was running using the Listener Control utility SET command are saved to the `listener.ora` file when the STOP command is issued. When you set the parameter to `false`, the Listener Control utility does not save the runtime configuration changes to the `listener.ora` file.

Default

false

Values

```
true | false
```

Example

```
SAVE_CONFIG_ON_STOP_listener=true
```

8.4.15 SERVICE_RATE_listener_name

The `SERVICE_RATE_listener_name` control parameter specifies incoming connection rate that is allowed per service for an instance.

Purpose

To specify incoming connection rate that is allowed per service for an instance.

Usage Notes

Any user-specified value greater than `0` sets the maximum limit on the number of new connections per service-instance handled by the proxy listener every second. Listener rejects connections after it reaches the maximum limit. Client side connection failure is reported with the "`TNS:listener: rate limit reached`" error.

Default

`0`

Example 8-7 Example

```
SERVICE_RATE=10
```

8.4.16 SSL_CIPHER_SUITES

Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).

Purpose

To control the combination of authentication, encryption, and data integrity algorithms used by TLS. By default, the strongest protocol and cipher are negotiated between the database client and server. Setting this parameter will override the default behavior. You must use this parameter only if you have internal security controls that dictate the usage of certain protocol versions.

Usage Notes

Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.

Enclose the `SSL_CIPHER_SUITES` parameter value in parentheses. Otherwise, the cipher suite setting does not parse correctly.

Default

None

Values

Approved ciphers compatible with TLS 1.3:

- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256` (non-FIPS only)
- `TLS_AES_128_CCM_SHA256`
- `TLS_AES_128_GCM_SHA256`

Approved ciphers compatible with TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

Deprecated ciphers compatible with TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA`
- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_DHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA`

Examples

```
SSL_CIPHER_SUITES=(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

```
SSL_CIPHER_SUITES=(TLS_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

Related Topics

- Set the TLS Cipher Suites on the Server
- Set the TLS Cipher Suites on the Client

8.4.17 SSL_CLIENT_AUTHENTICATION

Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).

Purpose

To enable client authentication in a TLS connection. The connection can be one-way or two-way (mutual TLS or mTLS).

Usage Notes

When set to `TRUE`, a two-way TLS connection is initiated. Both the client and server (including the listener) authenticate each other. For example, if you set this parameter to `TRUE` in the server configuration (server-side `sqlnet.ora`), then the server attempts to authenticate the client. If you set it to `TRUE` in the listener configuration (`listener.ora`), then the listener attempts to authenticate the client.

When set to `FALSE`, only the client authenticates the server and listener as a one-way TLS connection. For example, if you set this parameter to `FALSE` in the server configuration, then the server does not authenticate the client. If you set it to `FALSE` in the listener configuration, then the listener does not authenticate the client.

When set to `OPTIONAL`, the server behaves as follows:

- If the client sends a certificate, then the connection is completed as a two-way TLS connection after authenticating the client.
- If the client does not send a certificate, then the connection is completed as a one-way TLS connection.

Ensure that this parameter setting is consistent for the server or listener (on one side) and the client (on the other). Otherwise, the connection may fail. For example, if you enable client authentication in the server or listener configuration, then you must enable it in the client configuration.

Default

`TRUE`

Values

- `TRUE` | `ON` | `YES` | `1`: To enable mTLS

- `FALSE` | `OFF` | `NO` | `0`: To enable one-way TLS
- `OPTIONAL`: To enable both TLS and mTLS

Example

```
SSL_CLIENT_AUTHENTICATION=FALSE
```

Related Topics

- *Oracle Database Security Guide*

8.4.18 SSL_VERSION

Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.

Purpose

To define the version of TLS that must run on the systems with which the database server communicates. By default, the database server and client negotiate the strongest security protocol. Oracle does not recommend modifying this parameter, unless your security requirements mandate the usage of certain protocol versions.

Usage Notes

- Clients, listeners, and database servers must use compatible versions. Modify this parameter only when necessary to enforce the use of the more secure TLS protocol and not allow clients that only work with the older TLS protocols. The current default uses TLS 1.3, which is the version required for multiple security compliance requirements. If you need to specify TLS 1.2, then also include TLS 1.3 to allow more secure connections.
- In addition to `sqlnet.ora`, `listener.ora`, and `cman.ora`, you can specify this parameter under the `SECURITY` section of `tnsnames.ora` or directly as part of the connect string. The parameter value specified in the connect string takes precedence over the other specified values.
- Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

  In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.
- Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.

  SSLv3 is a much less secure protocol to secure the database server-to-client connection. Instead of using SSLv3, allow the database server and client to negotiate the most secure protocol that is common between the server and the client. Oracle Database 23ai provides TLS 1.2 and TLS 1.3 protocols for certificate-based network encryption.

- If you set `SSL_VERSION` to `undetermined`, then the most secure TLS protocol version is used. You can also use the `SSL_VERSION=undetermined` setting in the connect string for a specific connection to override the `SSL_VERSION` value configured in the `sqlnet.ora`, `listener.ora`, or `cman.ora` file.
- If you do not set `SSL_VERSION` to any value, then all the supported TLS protocol versions are tried starting with the most secure version. This is typically the most common configuration, ensuring that the strongest protocol is chosen during TLS negotiation.

Values

`undetermined | TLSv1.2 | TLSv1.3`

Default

`undetermined`

Syntax and Examples

- To specify a single protocol version:

  `SSL_VERSION=`*`TLS_protocol_version`*

  For example:

  ```
  SSL_VERSION=TLSv1.3
  ```

- To specify multiple protocol versions, use a comma-separated string of values, enclosed in parenthesis:

  `SSL_VERSION=(`*`TLS_protocol_version1`*`,`*`TLS_protocol_version2`*`)`

  For example:

  ```
  SSL_VERSION=(TLSv1.2,TLSv1.3)
  ```

> **Note:**
>
> Do not enclose protocol versions in parenthesis while specifying this parameter in the `tnsnames.ora` file or as part of the connect string, otherwise the setting will not parse correctly. For example:
>
> ```
> net_service_name=
>   (DESCRIPTION=
>      (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver)(PORT=1522))
>      (SECURITY=(SSL_VERSION=TLSv1.2,TLSv1.3))
>   )
> ```

Related Topics

- Set the Required TLS Version on the Server
- Set the Required TLS Version on the Client

8.4.19 SUBSCRIBE_FOR_NODE_DOWN_EVENT_*listener_name*

Purpose

To subscribe to Oracle Notification Service (ONS) notifications for downed events.

Usage Notes

By default, the listener subscribes to the ONS node down event on startup, if ONS is available. This subscription enables the listener to remove the affected service when it receives node down event notification from ONS. The listener uses asynchronous subscription for the event notification. Alter this behavior by setting `SUBSCRIBE_FOR_NODE_DOWN_EVENT_`*`listener_name`*`=off` in `listener.ora`.

Default

on

Values

`on | off`

8.4.20 USE_SID_AS_SERVICE_*listener_name*

Purpose

To enable the system identifier (SID) in the connect descriptor to be interpreted as a service name when a user attempts a database connection.

Usage Notes

Database clients with earlier releases of Oracle Database that have hard-coded connect descriptors can use this parameter to connect to a container or pluggable database.

For an Oracle container database, the client must specify a service name in order to connect to it. Setting this parameter to `on` instructs the listener to use the SID in the connect descriptor as a service name and connect the client to the specified database.

Default

off

Example

```
USE_SID_AS_SERVICE_listener=on
```

8.4.21 VALID_NODE_CHECKING_REGISTRATION_*listener_name*

The *listener.ora* control parameter `VALID_NODE_CHECKING_REGISTRATION_listener_name` determines if valid node checking registration is performed, or if the subnet is allowed.

Purpose

To determine whether valid node checking registration is performed, or the subnet is allowed.

Usage Notes

- When set to `on`, valid node checking registration is performed at the listener for any incoming registration request, and only local IP addresses are allowed.
- Starting with Oracle Grid Infrastructure 12c, for a SCAN listener, if the `VALID_NODE_CHECKING_REGISTRATION_listener_name` and `REGISTRATION_INVITED_NODES_listener_name` parameters are set in the `listener.ora` file, then the listener agent overwrites these parameters.

Default

on

Values

- `off | 0` to specify valid node checking registration is off, and no checking is performed.
- `on | 1 | local` to specify valid node checking registration is on, and all local IP addresses can register. If a list of invited nodes is set, then all IP addresses, host names, or subnets in the list as well as local IP addresses are allowed.
- `subnet | 2` to specify valid node checking registration is on, and all machines in the local subnets are allowed to register. If a list of invited nodes is set, then all nodes in the local subnets as well as all IP addresses, host names and subnets in the list are allowed.

Example

```
VALID_NODE_CHECKING_REGISTRATION_listener=on
```

> **See Also:**
>
> *Oracle Real Application Clusters Administration and Deployment Guide* for information about valid node checking for registration

8.4.22 WALLET_LOCATION

Use the `WALLET_LOCATION` parameter to specify the location of Oracle wallets.

Purpose

To specify the directory path where you want to create and store an Oracle wallet. Wallets securely contain certificates, secrets, private keys, and trust points used by Oracle Database.

Usage Notes

- Deprecation of the server-side setting:

  The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

  For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.

- Where to set this parameter:

  You can set `WALLET_LOCATION` in the `sqlnet.ora` file to specify a common wallet location for all connections. You can also set it in the connect string or `tnsnames.ora` file to specify a different wallet location for a particular connection.

  Use of `WALLET_LOCATION` in the connect string or `tnsnames.ora` overrides the `sqlnet.ora` `WALLET_LOCATION` setting for the specific `tnsnames.ora` service. The `tnsnames.ora` `WALLET_LOCATION` setting enables a client to initiate multiple TLS sessions using different TLS certificates in the same client process.

- Setting to use the system default certificate store instead of a client-side wallet:

  The Linux and Windows database clients can use the system default certificate store to validate the Oracle Database server certificate, instead of creating a local wallet with root certificate. The default certificate store is located in `/etc/pki/tls/cert.pem` on Linux and Microsoft Certificate Store (MCS) on Windows.

  If you set `WALLET_LOCATION=SYSTEM` in the connect string (in `tnsnames.ora` or directly to the command line), then the database client uses the default certificate store to validate the server certificate. In this case, the server certificate needs to be signed by a trusted root certificate that is already installed in the default certificate store.

  For example:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1234))
       (SECURITY=(WALLET_LOCATION=SYSTEM))
       (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
     )
```

- Order in which the database client searches for a client wallet:

  1. The database client first tries to use a wallet from the `WALLET_LOCATION` directory specified in the connect string.
  2. If no wallet is present, then the client searches for the `WALLET_LOCATION` parameter value in the `sqlnet.ora` file.

3. If no wallet is present, then the client searches for a wallet in the `$TNS_ADMIN` environment variable directory.
4. If no wallet is present, then the client searches in the default wallet location, that is, `/etc/ORACLE/WALLETS/username` on Linux and `C:\Users\username\\ORACLE\WALLETS` on Windows.
5. If no wallet is present, then the client uses the wallet from the system default certificate store.

You can specify `WALLET_LOCATION` as `SYSTEM` in the connect string to ignore all the wallet configurations and use the system default certificate store instead.

- Setting for walletless TLS connections:
  The `WALLET_LOCATION` parameter is optional for TLS connections that do not use a client wallet. If you do not include `WALLET_LOCATION` in the connect string, `tnsnames.ora`, or `sqlnet.ora`, then the driver automatically picks up common root certificates from the system default certificate store (if the system is Windows or Linux).

  However, you may need to perform additional steps in the following cases:

  – If `WALLET_LOCATION` is set in `sqlnet.ora` for all connections, then you can override this setting for a specific connection that does not need a client wallet (using `WALLET_LOCATION=SYSTEM` in the connect string).
  – If a wallet is present in the `$TNS_ADMIN` environment variable directory, then the database client uses the `$TNS_ADMIN` path as the default wallet location. In this case, you can either override the `WALLET_LOCATION` setting (using `WALLET_LOCATION=SYSTEM` in the connect string) or remove that wallet.

- Storage of wallet files:
  The password-protected wallet is stored in an `ewallet.p12` file. The auto-login and local auto-login wallets are stored in a `cwallet.sso` file.

  For example, if an Oracle wallet is stored in the Microsoft Windows registry and the wallet's key `(KEY)` is `SALESAPP`, then the storage location of the password-protected wallet is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\EWALLET.P12`. The storage location of the auto-login and local auto-login wallets is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\CWALLET.SSO`.

Additional Parameters

Use `SOURCE` to specify the type of storage and storage location for wallets, as follows:

- `METHOD`: Type of storage
- `METHOD_DATA`: Storage location:
  – `DIRECTORY`: Location of wallet on the file system
  – `KEY`: Wallet type and location in the Microsoft Windows registry

Syntax and Examples

The syntax depends on the wallet as follows:

- Wallet on the file system:

```
WALLET_LOCATION=
  (SOURCE=
```

```
   (METHOD=file)
   (METHOD_DATA=
      (DIRECTORY=directory)))
```

For example:

```
WALLET_LOCATION=
  (SOURCE=
      (METHOD=file)
      (METHOD_DATA=
         (DIRECTORY=/etc/oracle/wallets/databases)))
```

- Microsoft certificate store:

```
WALLET_LOCATION=
  (SOURCE=
     (METHOD=mcs))
```

The key-value pair for MCS omits the `METHOD_DATA` parameter because MCS does not use wallets. Instead, Oracle PKI (public key infrastructure) applications obtain certificates, trust points and private keys directly from a user's profile.

- Wallet in the Microsoft Windows registry:

```
WALLET_LOCATION=
   (SOURCE=
      (METHOD=reg)
      (METHOD_DATA=
         (KEY=registry_key)))
```

For example:

```
WALLET_LOCATION=
   (SOURCE=
     (METHOD=reg)
     (METHOD_DATA=
        (KEY=SALESAPP)))
```

Default

None

Related Topics

- *Oracle Database Security Guide*

8.5 ADR Diagnostic Parameters for Oracle Net Listener

The diagnostic data for the critical errors is quickly captured and stored in the ADR for Oracle Net listener.

Since Oracle Database 11g, Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems. The problems are critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

When a critical error occurs, it is assigned an incident number, and diagnostic data for the error, such as traces and dumps, are immediately captured and tagged with the incident number. The data is then stored in the Automatic Diagnostic Repository (ADR), a file-based repository outside the database.

This section includes the parameters used when ADR is enabled. ADR is enabled by default. Non-ADR parameters listed in the `listener.ora` file are ignored when ADR is enabled.

The following `listener.ora` parameters are used when ADR is enabled (when `DIAG_ADR_ENABLED` is set to `on`):

- ADR_BASE_listener_name
  The `ADR_BASE_listener_name` parameter is a diagnostic parameter specifies the base directory that stores tracing and logging incidents when ADR is enabled.
- DIAG_ADR_ENABLED_listener_name
  The `DIAG_ADR_ENABLED_listener_name` is a diagnostic parameter of the `listener.ora` file. It indicates whether ADR is enabled.
- LOG_FILE_NUM_listener_name
  The `LOG_FILE_NUM_listener_name` is a diagnostic parameter of the `listener.ora` file that specifies the number of log file segments.
- LOG_FILE_SIZE_listener_name
  The `LOG_FILE_SIZE_listener_name` diagnostic parameter of the`listener.ora` file specifies the size of each log file segment.
- LOGGING_listener_name
  The `LOGGING_listener_name` diagnostic parameter of the `listener.ora` file turns logging on or off.
- TRACE_LEVEL_listener_name
  The `TRACE_LEVEL_listener_name` diagnostic parameter of the `listener.ora` file turns listener tracing on, at a specific level, or turns it off.
- TRACE_TIMESTAMP_listener_name
  The `TRACE_TIMESTAMP_listener_name` diagnostic parameter of the `listener.ora` file adds a time stamp to every trace event in the trace file for the listener.

8.5.1 ADR_BASE_*listener_name*

The `ADR_BASE_listener_name` parameter is a diagnostic parameter specifies the base directory that stores tracing and logging incidents when ADR is enabled.

Purpose

To specify the base directory that stores tracing and logging incidents when ADR is enabled.

Default

The default is `ORACLE_BASE`, or `ORACLE_HOME/log` if `ORACLE_BASE` is not defined.

Values

Any valid directory path to a directory with write permission.

Example

```
ADR_BASE_listener=/oracle/network/trace
```

8.5.2 DIAG_ADR_ENABLED_*listener_name*

The `DIAG_ADR_ENABLED_listener_name` is a diagnostic parameter of the `listener.ora` file. It indicates whether ADR is enabled.

Purpose

To indicate whether ADR tracing is enabled.

Usage Notes

When the `DIAG_ADR_ENABLED_listener_name` parameter is set to `on`, then ADR file tracing is used. When the `DIAG_ADR_ENABLED_listener_name` parameter is set to `off`, then non-ADR file tracing is used.

Default

`on`

Values

`on|off`

Example 8-8 Example

```
DIAG_ADR_ENABLED_listener=on
```

8.5.3 LOG_FILE_NUM_*listener_name*

The `LOG_FILE_NUM_listener_name` is a diagnostic parameter of the `listener.ora` file that specifies the number of log file segments.

Purpose

To specify the number of log file segments. At any point of time there can be only *n* log file segments where *n* is `LOG_FILE_NUM_listener_name`. If the log grows beyond this number, then the older segments are deleted.

Default

No default. If you don't specify a value, or set the value to zero, then the number of segments grows indefinitely.

Values

Any integer value.

Example 8-9

```
LOG_FILE_NUM_listener=3
```

8.5.4 LOG_FILE_SIZE_*listener_name*

The `LOG_FILE_SIZE_listener_name` diagnostic parameter of the`listener.ora` file specifies the size of each log file segment.

Purpose

To specify the size of each log file segment. The size is in `MB`.

Default

`300 MB`

Values

Any integer value.

Example 8-10 Example

```
LOG_FILE_SIZE_listener=10
```

8.5.5 LOGGING_*listener_name*

The `LOGGING_listener_name` diagnostic parameter of the `listener.ora` file turns logging on or off.

Purpose

To turn logging on or off.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

on

Values

`on` | `off`

Example

```
LOGGING_listener=on
```

8.5.6 TRACE_LEVEL_*listener_name*

The `TRACE_LEVEL_listener_name` diagnostic parameter of the `listener.ora` file turns listener tracing on, at a specific level, or turns it off.

Purpose

To turn listener tracing on, at a specific level, or to turn it off.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

Default

`off | 0`

Values

- `off` or `0` for no trace output
- `user` or `4` for user trace information
- `admin` or `10` for administration trace information
- `support` or `16` for Oracle Support Services trace information

Example

```
TRACE_LEVEL_listener=admin
```

8.5.7 TRACE_TIMESTAMP_*listener_name*

The `TRACE_TIMESTAMP_listener_name` diagnostic parameter of the `listener.ora` file adds a time stamp to every trace event in the trace file for the listener.

Purpose

To add a time stamp in the form of `dd-mmm-yyyy hh:mi:ss:mil` to every trace event in the trace file for the listener.

Usage Notes

This parameter is used with the TRACE_LEVEL_listener_name parameter. This parameter is also applicable when non-ADR tracing is used.

Default

on

Values

- `on` | `true`
- `off` | `false`

Example

```
TRACE_TIMESTAMP_listener=true
```

8.6 Non-ADR Diagnostic Parameters for Oracle Net Listener

This section lists the parameters used when ADR is disabled. The default value of `DIAG_ADR_ENABLED_listener_name` is `on`. Therefore, the `DIAG_ADR_ENABLED_listener_name` parameter *must* explicitly be set to `off` to use non-ADR tracing.

8.6.1 LOG_DIRECTORY_*listener_name*

Purpose

To specify the destination directory of the listener log file.

Usage Notes

Use this parameter when ADR is not enabled.

Default

```
ORACLE_HOME/network/log
```

Example

```
LOG_DIRECTORY_listener=/oracle/network/admin/log
```

8.6.2 LOG_FILE_*listener_name*

Purpose

To specify the name of the log file for the listener.

Usage Notes

Use this parameter when ADR is not enabled.

Default

```
listener.log
```

Example

```
LOG_FILE_listener=list.log
```

8.6.3 TRACE_DIRECTORY_*listener_name*

Purpose

To specify the destination directory of the listener trace file.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_HOME/network/trace`

Example

```
TRACE_DIRECTORY_listener=/oracle/network/admin/trace
```

8.6.4 TRACE_FILE_*listener_name*

Purpose

To specify the name of the trace file for the listener.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`listener.trc`

Example

```
TRACE_FILE_listener=list.trc
```

8.6.5 TRACE_FILEAGE_*listener_name*

Purpose

To specify the maximum age of listener trace files in minutes.

Usage Notes

When the age limit is reached, the trace information is written to the next file. The number of files is specified with the TRACE_FILENO_listener_name parameter. Use this parameter when ADR is not enabled.

Default

Unlimited

This is the same as setting the parameter to `0`.

Example 8-11 Example

```
TRACE_FILEAGE_listener=60
```

8.6.6 TRACE_FILELEN_*listener_name*

Purpose

To specify the size of the listener trace files in kilobytes (KB).

Usage Notes

When the size is met, the trace information is written to the next file. The number of files is specified using the TRACE_FILENO_listener_name parameter. Use this parameter when ADR is not enabled.

Default

Unlimited

Example

```
TRACE_FILELEN_listener=100
```

8.6.7 TRACE_FILENO_*listener_name*

Purpose

To specify the number of trace files for listener tracing.

Usage Notes

When this parameter is set along with the TRACE_FILELEN_listener_name parameter, trace files are used in a cyclical fashion. The first file is filled first, then the second file, and so on. When the last file has been filled, the first file is re-used, and so on.

The trace file names are distinguished from one another by their sequence number. For example, if the default trace file of `listener.trc` is used, and this parameter is set to 3, then the trace files would be named `listener1.trc`, `listener2.trc` and `listener3.trc`.

In addition, trace events in the trace files are preceded by the sequence number of the file. Use this parameter when ADR is not enabled.

Default

1

Example

```
TRACE_FILENO_listener=3
```

8.7 Class of Secure Transports Parameters

The class of secure transports (COST) parameters specify a list of transports that are considered secure for administration and registration of a particular listener.

The COST parameters identify which transports are considered secure for that installation and whether the administration of a listener requires secure transports. Configuring these parameters is optional.

- SECURE_REGISTER_listener_name
- Using COST Parameters in Combination
- DYNAMIC_REGISTRATION_listener_name
  `DYNAMIC_REGISTRATION_listener_name` is a class of secure transports (COST) parameter and it enables or disables dynamic registration of a listener.
- SECURE_PROTOCOL_listener_name
- SECURE_CONTROL_listener_name

> **See Also:**
>
> *Oracle Database Net Services Administrator's Guide* for additional information about COST parameters and listener security

8.7.1 SECURE_REGISTER_*listener_name*

Purpose

To specify the transports on which registration requests are to be accepted.

Usage Notes

If the `SECURE_REGISTER_listener_name` parameter is configured with a list of transport names, then only the connections arriving on the specified transports are able to register the service with the listener. Connections arriving by other transport protocols are refused. The following is an example:

```
SECURE_REGISTER_listener1 = (TCPS,IPC)
```

In the preceding example, registration requests are accepted only on TCPS and IPC transports.

If no values are entered for this parameter, then the listener accepts registration requests from any transport.

Syntax

```
SECURE_REGISTER_listener_name =
[(]transport1[,transport2, ....,transportn)]
```

In the preceding example, `transport1`, `transport2`, and `transportn` are valid, installed transport protocol names.

If this parameter and SECURE_CONTROL_listener_name are configured, then they override the SECURE_PROTOCOL_listener_name parameter.

Example

```
LISTENER1=
 (DESCRIPTION=
```

```
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
      (ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=1522))))
  SECURE_REGISTER_listener1=tcps
```

8.7.2 Using COST Parameters in Combination

COST parameters can also be used in combination to further control which transports accept service registration and control commands.

In Example 8-12, control commands are accepted only on the IPC channel and the TCPS transport, and service registrations are accepted only on an IPC channel.

Example 8-12 Combining COST Parameters

```
LISTENER1=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
     (ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=1522))))
  SECURE_CONTROL_listener1=(tcps,ipc)
  SECURE_REGISTER_listener1=ipc
```

In Example 8-13, control commands are accepted only on the TCPS transport, and service registrations are accepted only on the IPC channel.

Example 8-13 Combining COST Parameters

```
LISTENER1=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
     (ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=1522))))
  SECURE_CONTROL_listener1=tcps
  SECURE_PROTOCOL_listener1=ipc
```

8.7.3 DYNAMIC_REGISTRATION_listener_name

`DYNAMIC_REGISTRATION_`*`listener_name`* is a class of secure transports (COST) parameter and it enables or disables dynamic registration of a listener.

Purpose

To enable or disable dynamic registration.

Usage Notes

Static registrations are not affected by this parameter.

Default

The default value is `on`. Unless this parameter is explicitly set to `off`, all registration connections are accepted.

Values

- `on`: The listener accepts dynamic registration.
- `off`: The listener refuses dynamic registration.

Example 8-14 Example

```
DYNAMIC_REGISTRATION_listener_name=on
```

8.7.4 SECURE_PROTOCOL_*listener_name*

Purpose

To specify the transports on which administration and registration requests are accepted.

Usage Notes

If this parameter is configured with a list of transport names, then the control commands and service registration can happen only if the connection belongs to the list of transports.

If this parameter is not present and neither SECURE_CONTROL_listener_name or SECURE_REGISTER_listener_name are configured, then all supported transports accept control and registration requests.

If the SECURE_CONTROL_listener_name and SECURE_REGISTER_listener_name parameters are configured, then they override the `SECURE_PROTOCOL_listener_name` parameter.

Syntax

```
SECURE_PROTOCOL_listener_name =
[()transport1[,transport2, ....,transportn)]
```

In the preceding syntax, `transport1`, `transport2`, and `transportn` are valid, installed transport protocol names.

Example

```
LISTENER1=
 (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
      (ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=1522))))
  SECURE_PROTOCOL_listener1=tcps
```

8.7.5 SECURE_CONTROL_*listener_name*

Purpose

To specify the transports on which control commands are to be serviced.

Usage Notes

If the `SECURE_CONTROL_listener_name` parameter is configured with a list of transport names, then the control commands are serviced only if the connection is one of the listed transports. Connections arriving by other transport protocols are refused. The following is an example:

```
SECURE_CONTROL_listener1 = (TCPS,IPC)
```

In the preceding example, administration requests are accepted only on TCPS and IPC transports.

If no values are entered for this parameter, then the listener accepts any connection on any endpoint.

Syntax

```
SECURE_CONTROL_listener_name =
[()]transport1[,transport2, ....,transportn)]
```

In the preceding syntax, `transport1`, `transport2`, and `transportn` are valid, installed transport protocol names.

Example

```
LISTENER1=
 (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
      (ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=1522))))
  SECURE_CONTROL_LISTENER1=tcps
```

9

Oracle Connection Manager Parameters

This chapter provides a complete listing of the `cman.ora` file configuration parameters.

- Overview of Oracle Connection Manager Configuration File
  Oracle Connection Manager configuration information is stored in the `cman.ora` file.
- Oracle Connection Manager Parameters
  This section lists and describes the following `cman.ora` file parameters:
- Oracle Connection Manager in Traffic Director Mode Parameters
- ADR Diagnostic Parameters for Oracle Connection Manager
  The diagnostic data for critical errors is quickly captured and stored in the ADR for Oracle Connection Manager.
- Non-ADR Diagnostic Parameters for Oracle Connection Manager
  This section lists the parameters used when ADR is disabled:
- Oracle Connection Manager Tunneling Parameters
  This section lists the parameters that you must configure to enable tunneling.

9.1 Overview of Oracle Connection Manager Configuration File

Oracle Connection Manager configuration information is stored in the `cman.ora` file.

Oracle Connection Manager Configuration File

Oracle Connection Manager configuration information consists of the following elements:

- Protocol address of the Oracle Connection Manager listener
- Access control parameters
- Performance parameters

By default, the `cman.ora` file is located in the *`ORACLE_HOME`*`/network/admin` directory. You can also store the `cman.ora` file in the following locations:

- The directory specified by the `TNS_ADMIN` environment variable or registry value.
- On Linux and UNIX operating systems, the global configuration directory. For example, on the Oracle Solaris operating system, this directory is `/var/opt/oracle`.
- *`ORACLE_BASE_HOME`*`/network/admin` directory.
- *`ORACLE_HOME`*`/network/admin` directory.

Example 9-1 Sample cman.ora File

```
CMAN=
  (CONFIGURATION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
    (RULE_LIST=
      (RULE=(SRC=192.0.2.32/27)(DST=sales-server)(SRV=*)(ACT=accept))
```

```
    (ACTION_LIST=(AUT=on)(MCT=120)(MIT=30)))
  (RULE=(SRC=foo)(DST=hr-server)(SRV=cmon)(ACT=accept)))
 (PARAMETER_LIST=
   (MAX_GATEWAY_PROCESSES=8)
   (MIN_GATEWAY_PRCESSSES=3)
   (DIAG_ADR_ENABLED=ON)
   (ADR_BASE=/oracle/log)))
```

cman.ora File Sections

- Listening address: Preceded by `ADDRESS=`, this section contains information pertinent to the listener. The `ADDRESS` parameter is required.
- Rule list: Preceded by `RULE_LIST=`, this section contains rule information. The`RULE` parameter is listed in the rule list section of the file. The `RULE` parameter is required.
- Rule Group: Preceded by `RULE_GROUP=`, this section contains `rule_list` grouped by service names. You can use either the `rule_group` syntax or the `rule_list` syntax.
- Parameter list: Preceded by `PARAMETER_LIST=`, this section contains all other parameters including those listed in " ADR Diagnostic Parameters for Oracle Connection Manager", and "Non-ADR Diagnostic Parameters for Oracle Connection Manager".

  The following parameters are allowed in the parameter list section of the `cman.ora` file. The default values are bold. To override the default setting for a parameter, enter the parameter and a nondefault value.

  `ASO_AUTHENTICATION_FILTER={`**`off`**` | on}`

  `CONNECTION_STATISTICS={`**`no`**` | yes}`

  `EVENT_GROUP={init_and_term | memory_ops | conn_hdlg | proc_mgmt | reg_and_load | wake_up | timer | cmd_proc | relay}`

  `IDLE_TIMEOUT=`**`0`** or greater

  `INBOUND_CONNECT_TIMEOUT=0` or greater. The default value is 60.

  `LOG_DIRECTORY=`*`log_directory`*. The default value is `ORACLE_HOME/network/log`.

  `LOG_LEVEL={off | user | admin |` **`support`**`}`

  `MAX_CMCTL_SESSIONS=` Any positive number. The default value is 4.

  `MAX_CONNECTIONS=` A value between 1 and 1024. The default value is 256.

  `MAX_GATEWAY_PROCESSES=` Any number greater than the minimum number of gateway processes up to 64. The default value is 16.

  `MIN_GATEWAY_PROCESSES=` Any positive number less than or equal to 64. Must be less than or equal to the maximum number of gateway processes. The default value is 2.

  `OUTBOUND_CONNECT_TIMEOUT=`**`0`** or greater

  `SESSION_TIMEOUT=`**`0`** or greater

  `TRACE_DIRECTORY=`*`trace_directory`*. The default value is `ORACLE_HOME/network/trace`.

TRACE_FILELEN= Any positive number. The default value is 0 (zero).

TRACE_FILENO= Any positive number. The default value is 0 (zero).

TRACE_LEVEL={**off** | user | admin | support}

TRACE_TIMESTAMP={**off** | on}

```
(PARAMETER_LIST=
      (ASO_AUTHENTICATION_FILTER=ON)
      (CONNECTION_STATISTICS=NO)
      (EVENT_GROUP=INIT_AND_TERM,MEMORY_OPS,PROCESS_MGMT)
      (IDLE_TIMEOUT=30)
      (INBOUND_CONNECT_TIMEOUT=30)
      (LOG_DIRECTORY=/home/user/network/admin/log)
      (LOG_LEVEL=SUPPORT)
      (MAX_CMCTL_SESSIONS=6)
      (MAX_CONNECTIONS=512)
      (MAX_GATEWAY_PROCESSES=10)
      (MIN_GATEWAY_PROCESSES=4)
      (OUTBOUND_CONNECT_TIMEOUT=30)
      (SESSION_TIMEOUT=60)
      (TRACE_DIRECTORY=/home/user/network/admin/trace)
      (TRACE_FILELEN=100)
      (TRACE_FILENO=2)
      (TRACE_LEVEL=SUPPORT)
      (TRACE_TIMESTAMP=ON)
      (VALID_NODE_CHECKING_REGISTRATION=ON)
      (REGISTRATION_EXCLUDED_NODES = 10.1.26.*)
      (REGISTRATION_INVITED_NODES = 10.1.35.*)
)
```

9.2 Oracle Connection Manager Parameters

This section lists and describes the following `cman.ora` file parameters:

- ADDRESS
  The `ADDRESS` networking parameter specifies the protocol address of Oracle Connection Manager.
- ASO_AUTHENTICATION_FILTER
  It is a networking parameter for Oracle Connection Manager. It instructs Oracle Connection Manager to check the connection requests for Secure Network Services (SNS).
- BANDWIDTH
  Use the `BANDWIDTH` parameter to limit all the connections of a service to a specified value in bytes per second.
- CLIENT_DN_RULE_MATCH
  Use this parameter to enable filtering of Transport Layer Security (TLS) connections using `DN_LIST` in `RULE_GROUP`.
- COMPRESSION

- COMPRESSION_LEVELS
  The `COMPRESSION_LEVELS` networking parameter of the `cman.ora` file specifies the CPU usage and compression ratio.
- COMPRESSION_THRESHOLD
- CONNECTION_STATISTICS
  `CONNECTION_STATISTICS` networking parameter of the `cman.ora` file specifies whether the `SHOW_CONNECTIONS` command displays connection statistics.
- DN_LIST
  Use this parameter to specify a list of common names (CN) that are allowed to connect to a service using Transport Layer Security (TLS).
- ENABLE_IP_FORWARDING
  Use the `cman.ora` parameter `ENABLE_IP_FORWARDING` to forward client IP address to the database server.
- EVENT_GROUP
  `EVENT_GROUP` networking parameter of the `cman.ora` file specifies which event groups are logged.
- EXPIRE_TIME
  The `EXPIRE_TIME` networking parameter of `cman.ora` file specifies a time interval, in minutes, to send a check to verify that client/gateway connections are active.
- GROUP
  Use the `GROUP` parameter to specify a `rule_list` for a service.
- IDLE_TIMEOUT
- INBOUND_CONNECT_TIMEOUT
- IP_RATE_COUNT
  The `IP_RATE_COUNT` parameter of the `cman.ora` file specifies the maximum number of client connections allowed from an IP address in the specified time interval.
- IP_RATE_INTERVAL
  The `IP_RATE_INTERVAL` parameter of the `cman.ora` file specifies the number of seconds for which Oracle Connection Manager accepts new connections from a single IP address.
- IP_RATE_BLOCK
  The `IP_RATE_BLOCK` parameter of the `cman.ora` file specifies the time duration, in minutes, for which an IP address is blocked after exceeding the defined IP rate limit.
- LOG_DIRECTORY
- LOG_FILE_NUM
  `LOG_FILE_NUM` networking parameter of the `cman.ora` file specifies the number of log file segments.
- LOG_FILE_SIZE
  `LOG_FILE_SIZE` networking parameter of the`cman.ora` file specifies the size of each log file segment.
- LOG_LEVEL
- LOG_SUPPRESS_NODES
  Use the `cman.ora` parameter `LOG_SUPPRESS_NODES` to specify the addresses for which you want to disable logging of health check errors in the Oracle Connection Manager (CMAN) log file.

- MAX_ALL_CONNECTIONS
- MAX_CMCTL_SESSIONS
- MAX_BANDWIDTH_GROUP
  Use the `MAX_BANDWIDTH_GROUP` parameter to specify the maximum number of services that can be configured.
- MAX_CONNECTIONS
- MAX_GATEWAY_PROCESSES
- MAX_REG_CONNECTIONS
- MIN_GATEWAY_PROCESSES
- NEXT_HOP
  The `NEXT_HOP` parameter provides static routing of client connections from Oracle Connection Manager (Oracle CMAN).
- OUTBOUND_CONNECT_TIMEOUT
- REGISTRATION_EXCLUDED_NODES
  The Oracle Connection Manager parameter file (`cman.ora`) REGISTRATION_EXCLUDED_NODES specifies the list of nodes that cannot register with the listener.
- REGISTRATION_INVITED_NODES
  The Oracle Connection Manager parameter file (`cman.ora`) `REGISTRATION_EXCLUDED_NODES` parameter specifies the list of node that can register with the listener.
- REST_ADDRESS
  Use the `REST_ADDRESS` parameter to configure REST endpoint hostname and port. Oracle CMAN listens to `tcps` endpoint based on the specified hostname and port.
- RULE
- SDU
- SERVICE_RATE
  The SERVICE_RATE parameter of cman.ora file specifies incoming connection rate that is allowed per service for an instance.
- SESSION_TIMEOUT
- SSL_CIPHER_SUITES
  Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).
- SSL_CLIENT_AUTHENTICATION
  Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).
- SSL_VERSION
  Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.
- TRACE_FILE
- TRACE_FILELEN
- TRACE_FILENO
- TRACE_LEVEL

- TRACE_TIMESTAMP
- USE_SERVICE_AS_TNSNAMES_ALIAS
  Use this parameter for static routing of client connections from Oracle connection manager based on client's service name.
- USE_SID_AS_SERVICE
  The `USE_SID_AS_SERVICE` Oracle Connection Manager parameter enables the system identifier (SID) in the connect descriptor to be interpreted as a service name when a user attempts a database connection.
- VALID_NODE_CHECKING_REGISTRATION
- WALLET_LOCATION
  Use the `WALLET_LOCATION` parameter to specify the location of Oracle wallets.

9.2.1 ADDRESS

The `ADDRESS` networking parameter specifies the protocol address of Oracle Connection Manager.

Purpose

To specify the protocol address of Oracle Connection Manager.

Syntax

```
(ADDRESS=(PROTOCOL=protocol)(HOST=host_name)(PORT=port_number)
```

Usage Notes

You can tag Oracle Connection Manager addresses as admin endpoints. This is helpful in cases where you do not want to close all listening endpoints, so that Oracle Connection Manager Control utility continues to run admin commands using tagged listening endpoints. To do so, set the value of the `ADMIN` parameter to `YES` using the following syntax:

```
(ADDRESS=(PROTOCOL=protocol)(HOST=host_name)(PORT=port_number)
(ADMIN=YES))
```

Example

```
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
```

Related Topics

- CLOSE NON_ADMIN_ENDPOINTS
  Use the Oracle Connection Manager Control Utility command `CLOSE NON_ADMIN_ENDPOINTS` to close all non-admin listening endpoints.

9.2.2 ASO_AUTHENTICATION_FILTER

It is a networking parameter for Oracle Connection Manager. It instructs Oracle Connection Manager to check the connection requests for Secure Network Services (SNS).

Purpose

To specify whether Oracle Database security authentication settings must be used by the client.

Usage Notes

The global setting can be overridden by a rule-level setting in `ACTION_LIST`.

Values

- `on` to instruct Oracle Connection Manager to reject connection requests that are not using Secure Network Services (SNS). SNS is part of Oracle Database security.
- `off` to instruct Oracle Connection Manager not to check for SNS between the client and server. This is the default.

9.2.3 BANDWIDTH

Use the `BANDWIDTH` parameter to limit all the connections of a service to a specified value in bytes per second.

Usage Notes

Specify a limit on the number of bytes transmitted per second. You must include this parameter in the `parameter_list` section of the `cman.ora` file.

Example

```
BANDWIDTH = 524288
```

9.2.4 CLIENT_DN_RULE_MATCH

Use this parameter to enable filtering of Transport Layer Security (TLS) connections using `DN_LIST` in `RULE_GROUP`.

Purpose

A TLS connection is allowed only if there is a `GROUP` specified in `RULE_GROUP` for the requested service. This `GROUP` must be configured with `DN_LIST`.

Values

`ON`, `OFF`. By default the value is set to `OFF`.

Example

```
CLIENT_DN_RULE_MATCH=ON
```

9.2.5 COMPRESSION

Purpose

To enable or disable data compression. If both the Oracle Connection Manager and the other end (server or client or Oracle Connection Manager) have this parameter set to `ON`, then compression is used for the connection.

Default

`off`

Values

- `on` to enable data compression.
- `off` to disable data compression.

Example

```
COMPRESSION=on
```

9.2.6 COMPRESSION_LEVELS

The `COMPRESSION_LEVELS` networking parameter of the `cman.ora` file specifies the CPU usage and compression ratio.

Purpose

To specify the compression level.

Usage Notes

The compression levels are used at the time of negotiation to verify which levels are used at both ends, and select one level.

Default

`low`

Values

- `low` for low CPU usage and a low compression ratio.
- `high` for high CPU usage and a high compression ratio.

Example 9-2 Example

```
COMPRESSION_LEVELS=high,low
```

9.2.7 COMPRESSION_THRESHOLD

Purpose

To specify the minimum data size, in bytes, for which compression is required.

Usage Notes

Compression is not be done if the size of the data to be sent is less than this value.

Default

`1024 bytes`

Example

```
COMPRESSION_THRESHOLD=1024
```

9.2.8 CONNECTION_STATISTICS

`CONNECTION_STATISTICS` networking parameter of the `cman.ora` file specifies whether the `SHOW_CONNECTIONS` command displays connection statistics.

Purpose

To specify whether the `SHOW_CONNECTIONS` command displays connection statistics.

Usage Notes

The global setting can be overridden by a rule-level setting in `ACTION_LIST`.

Values

- `yes` to display statistics.
- `no` to not display statistics. This is the default.

9.2.9 DN_LIST

Use this parameter to specify a list of common names (CN) that are allowed to connect to a service using Transport Layer Security (TLS).

Purpose

An incoming TLS connection is allowed only if the string provided in common name (CN) of the distinguished name (DN) matches with at least one value in the list of values provided in the `DN_LIST` parameter.

Usage Notes

`DN_LIST` is a comma separated list of common names. The values in the `DN_LIST` parameter is matched only when the `client_dn_rule_match` parameter is set to `ON`.

You must configure `DN_LIST` inside `DESCRIPTION` of the `GROUP` parameter.

Example

```
(GROUP =
     (DESCRIPTION = (NAME = service_name)(DN_LIST = phx,blr))
     (RULE_LIST =
```

```
      (RULE=...)
)
```

9.2.10 ENABLE_IP_FORWARDING

Use the `cman.ora` parameter `ENABLE_IP_FORWARDING` to forward client IP address to the database server.

Purpose

When set to `ON`, Oracle Connection Manager (CMAN) forwards the client source address as seen by it to the database server.

Usage Notes

In addition to the `ENABLE_IP_FORWARDING` parameter, you must set the `TCP.ALLOWED_PROXIES` parameter in the server-side `sqlnet.ora` file. The `TCP.ALLOWED_PROXIES` parameter specifies a list of the CMAN instances that can forward client address.

You can use the `SYS_CONTEXT ('USERENV','IP_ADDRESS')` function to query the forwarded client address details.

Values

- `ON` | `TRUE` | `YES` | `1`: To enable client address forwarding
- `OFF` | `FALSE` | `NO` | `0`: To disable client address forwarding

Default

`OFF`

Example

```
ENABLE_IP_FORWARDING=ON
```

Related Topics

- TCP.ALLOWED_PROXIES
  Use the `sqlnet.ora` parameter `TCP.ALLOWED_PROXIES` to specify a list of the Oracle Connection Manager (CMAN) addresses that can forward client IP address to the database server.
- *Oracle Database SQL Language Reference*

9.2.11 EVENT_GROUP

`EVENT_GROUP` networking parameter of the `cman.ora` file specifies which event groups are logged.

Purpose

To specify which event groups are logged.

Usage Notes

Multiple events may be designated using a comma-delimited list.

Values

- `alert` for alert notifications.
- `cmd_proc` for command processing.
- `conn_hdlg` for connection handling.
- `init_and_term` for initialization and termination.
- `memory_ops` for memory operations.
- `proc_mgmt` for process management.
- `reg_and_load` for registration and load update.
- `relay` for events associated with connection control blocks.
- `timer` for gateway timeouts.
- `wake_up` for events related to Connection Manager Administration (CMADMIN) wake-up queue.

> **Note:**
>
> The event group `ALERT` cannot be turned off.

9.2.12 EXPIRE_TIME

The `EXPIRE_TIME` networking parameter of `cman.ora` file specifies a time interval, in minutes, to send a check to verify that client/gateway connections are active.

Purpose

To specify a time interval, in minutes, to send a check to verify that client/server connections are active.

Usage Notes

Setting a value greater than `0` ensures that connections are not left open indefinitely, due to an unusual client termination. If the system supports TCP keepalive tuning, then Oracle Net Services automatically uses the enhanced detection model, and tunes the TCP keepalive parameters

If the probe finds a terminated connection, or a connection that is no longer in use, then it returns an error, causing the server process to exit.

This parameter is primarily intended for the database server, which typically handles multiple connections at any one time.

Limitations on using this terminated connection detection feature are:

- It is not allowed on bequeathed connections.

- Though very small, a probe packet generates additional traffic that may downgrade network performance.
- Depending on which operating system is in use, the server may need to perform additional processing to distinguish the connection probing event from other events that occur. This can also result in degraded network performance.

Values

- `0`: To disable terminated connection detection.
- Any number greater than `0`: To enable terminated connection detection. The number equals the time interval in minutes.

Default

`0`

Example 9-3 Example

```
EXPIRE_TIME=10
```

9.2.13 GROUP

Use the `GROUP` parameter to specify a `rule_list` for a service.

Purpose

This parameter is listed in the `RULE_GROUP` section of the `cman.ora` file preceded by `RULE_GROUP=`.

Syntax

```
(GROUP =
      (DESCRIPTION = (NAME = service_name))
      (RULE_LIST =
        (RULE=...)
      )
```

Usage Notes

The service name `(SRV =)` in the rule should match the `service_name` specified in the `NAME` parameter. Alternatively, you can specify the service name using an asterisk `*`.

You can configure a `DEFAULT_GROUP` in `RULE_GROUP`. The rules that you specify in this section applies to those services that do not have an explicit `GROUP`. You do not need to specify `DESCRIPTION` inside a `DEFAULT_GROUP`.

Example

```
(RULE_GROUP=
     (GROUP =
       (DESCRIPTION = (NAME = sales.us.example.com))
       (RULE_LIST =
         (RULE=
         (SRC=client1-pc)
```

```
      (DST=sales-server)
      (SRV=*)
      (ACT=reject))
    )
  )
  (GROUP =
    (DESCRIPTION = (NAME = hr.us.example.com))
    (RULE_LIST =
      (RULE=
        (SRC=192.0.2.45)
        (DST=192.0.2.200)
        (SRV=*)
        (ACT=accept))
    )
  )
  (DEFAULT_GROUP =
    (RULE_LIST=
      (RULE=(SRC=*)(DST=*)(SRV=cmon)(ACT=accept)))
   )
)
```

9.2.14 IDLE_TIMEOUT

Purpose

To specify the amount of time that an established connection can remain active without transmitting data.

Usage Notes

The global setting can be overridden by a rule-level setting in `ACTION_LIST`.

Values

- `0` to disable the timeout. This is the default.
- Any number greater than 0 to enable the timeout. The number equals the timeout period in seconds.

9.2.15 INBOUND_CONNECT_TIMEOUT

Purpose

To specify how long in seconds the Oracle Connection Manager listener waits for a valid connection from a client or another instance of Oracle Connection Manager.

Values

- `60 sec` is the default. Use value `0` to disable timeout.
- Any number greater than 0 to enable the timeout. The number equals the timeout period in seconds.

9.2.16 IP_RATE_COUNT

The IP_RATE_COUNT parameter of the cman.ora file specifies the maximum number of client connections allowed from an IP address in the specified time interval.

Purpose

To enforce IP rate limit on the number of client connections allowed to Oracle Connection Manager (CMAN) from a single IP address. This security feature enables you to protect your database against potential denial-of-service (DoS) attacks.

Usage Notes

Use the IP_RATE_COUNT parameter under the PARAMETER_LIST section of the CMAN configuration. When set to a value greater than 1, the specified IP rate limit is enforced at the CMAN endpoint level.

You can use this parameter along with the optional IP_RATE_INTERVAL and IP_RATE_BLOCK parameters. IP_RATE_INTERVAL allows you to specify the number of seconds for which IP_RATE_COUNT connections are accepted. IP_RATE_BLOCK allows you to specify the duration for which the IP address is blocked after exceeding the defined IP_RATE_COUNT per IP_RATE_INTERVAL limit.

Default

None

Value

Any number greater than 1

Example

```
CMAN=
    (CONFIGURATION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
        (PARAMETER_LIST=(IP_RATE_COUNT=512)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- IP_RATE_INTERVAL
  The IP_RATE_INTERVAL parameter of the cman.ora file specifies the number of seconds for which Oracle Connection Manager accepts new connections from a single IP address.
- IP_RATE_BLOCK
  The IP_RATE_BLOCK parameter of the cman.ora file specifies the time duration, in minutes, for which an IP address is blocked after exceeding the defined IP rate limit.

9.2.17 IP_RATE_INTERVAL

The IP_RATE_INTERVAL parameter of the cman.ora file specifies the number of seconds for which Oracle Connection Manager accepts new connections from a single IP address.

Purpose

To specify the number of seconds for which IP_RATE_COUNT connections are accepted. This security feature enforces IP rate limit on client connections and thus protects your database against potential denial-of-service (DoS) attacks.

Usage Notes

This is an optional parameter. You can use it under the PARAMETER_LIST section along with the IP_RATE_COUNT parameter. IP_RATE_COUNT allows you to specify the number of connections allowed from an IP address.

You can also set the optional IP_RATE_BLOCK parameter to specify the duration for which the IP address is blocked after exceeding the defined IP_RATE_COUNT per IP_RATE_INTERVAL limit.

Default

1 second

Value

Any number less than or equal to 60. The number equals the time duration in seconds.

Example

```
CMAN=
    (CONFIGURATION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
        (PARAMETER_LIST=(IP_RATE_INTERVAL=5)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- IP_RATE_COUNT
  The IP_RATE_COUNT parameter of the cman.ora file specifies the maximum number of client connections allowed from an IP address in the specified time interval.
- IP_RATE_BLOCK
  The IP_RATE_BLOCK parameter of the cman.ora file specifies the time duration, in minutes, for which an IP address is blocked after exceeding the defined IP rate limit.

9.2.18 IP_RATE_BLOCK

The IP_RATE_BLOCK parameter of the cman.ora file specifies the time duration, in minutes, for which an IP address is blocked after exceeding the defined IP rate limit.

Purpose

To specify the time duration for which an IP address is blocked from establishing new connections. This security feature enforces IP rate limit on client connections and thus protects your database against potential denial-of-service (DoS) attacks.

Usage Notes

This is an optional parameter. You can use it under the PARAMETER_LIST section along with the IP_RATE_COUNT parameter. IP_RATE_COUNT allows you to specify the number of connections allowed from an IP address.

You can also set the optional IP_RATE_INTERVAL parameter to specify the number of seconds for which IP_RATE_COUNT connections are accepted. The IP address is blocked after exceeding the defined IP_RATE_COUNT per IP_RATE_INTERVAL limit.

Default

15 minutes

Value

Any number greater than 0. The number equals the time duration in minutes.

Example

```
CMAN=
    (CONFIGURATION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
         (PARAMETER_LIST=(IP_RATE_BLOCK=30)))
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- IP_RATE_COUNT
  The IP_RATE_COUNT parameter of the cman.ora file specifies the maximum number of client connections allowed from an IP address in the specified time interval.
- IP_RATE_INTERVAL
  The IP_RATE_INTERVAL parameter of the cman.ora file specifies the number of seconds for which Oracle Connection Manager accepts new connections from a single IP address.

9.2.19 LOG_DIRECTORY

Purpose

To specify the directory for the Oracle Connection Manager log files.

Default

ORACLE_BASE_*HOME*/network/log

9.2.20 LOG_FILE_NUM

LOG_FILE_NUM networking parameter of the cman.ora file specifies the number of log file segments.

Purpose

To specify the number of log file segments. At any point of time there can be only n log file segments where n is LOG_FILE_NUM and if the log grows beyond this number, then the older segments are deleted.

Default

No default. Number of segments grow indefinitely, if not specified or set to zero.

Values

Any integer value.

Example 9-4 Example

```
LOG_FILE_NUM=3
```

9.2.21 LOG_FILE_SIZE

LOG_FILE_SIZE networking parameter of thecman.ora file specifies the size of each log file segment.

Purpose

To specify the size of each log file segment. The size is in MB.

Default

300 MB

Values

Any integer value.

Example 9-5 Example

```
LOG_FILE_SIZE=10
```

9.2.22 LOG_LEVEL

Purpose

To specify the level for log messages.

Values

- `off` for no logging. This is the default.
- `user` for user-induced errors log information.
- `admin` for administration log information, such as installation-specific.
- `support` for Oracle Support Services information.

9.2.23 LOG_SUPPRESS_NODES

Use the `cman.ora` parameter `LOG_SUPPRESS_NODES` to specify the addresses for which you want to disable logging of health check errors in the Oracle Connection Manager (CMAN) log file.

Purpose

A CMAN frontend component, such as a load balancer, may perform periodic health checks by connecting to CMAN endpoint at the backend followed by immediately disconnecting from it. These health check operations generate error entries in the CMAN log file and are logged as connect failures. You can set this parameter to disable logging of such errors.

Usage Notes

Set this parameter in the `PARAMETER_LIST` section of the `cman.ora` file.

The list of addresses can include host names or CIDR notation for IPv4 and IPv6 addresses. The wildcard format (`*`) is supported for IPv4 addresses.

The presence of a host name in the list results in the inclusion of all IP addresses mapped to the host name. The host name must be consistent with the public network interface.

Value

```
LOG_SUPPRESS_NODES=(list of load balancer addresses)
```

`list of load balancer addresses` specifies valid nodes, subnet IP addresses, or names for which you want to disable logging.

Default

None

Example

```
LOG_SUPPRESS_NODES=(10.1.35.*, 10.1.34.0/24, 2001:DB8:fe38:7303, node1)
```

9.2.24 MAX_ALL_CONNECTIONS

Purpose

To specify the maximum number of concurrent registration and client connection sessions that can be supported by Oracle Connection Manager.

Usage Notes

This number includes registration connections from databases, and ongoing client connection establishment requests. After a connection is established, the clients do not maintain a connection to the listener. This limit only applies to client connections that are in the initial connection establishment phase from a listener perspective.

Default

Operating system-specific

Example

```
MAX_ALL_CONNECTIONS=40
```

9.2.25 MAX_CMCTL_SESSIONS

Purpose

To specify the maximum number of concurrent local or remote sessions of the Oracle Connection Manager control utility allowable for a given instance.

Usage Notes

One of the sessions must be a local session.

Values

Any number of sessions can be designated.

9.2.26 MAX_BANDWIDTH_GROUP

Use the `MAX_BANDWIDTH_GROUP` parameter to specify the maximum number of services that can be configured.

Usage Notes

Configure this parameter to a value of maximum services that your system supports. Add this parameter in the `parameter` section of the `cman.ora` file.

You can also configure this parameter with an additional 20% to 100% buffer, depending upon how often the services are created and destroyed in the system.

Example

```
MAX_BANDWIDTH_GROUP = 100
```

9.2.27 MAX_CONNECTIONS

Purpose

To specify the maximum number of connection slots that a gateway process can handle.

Values

Any number in the range of 1 to 1024.

9.2.28 MAX_GATEWAY_PROCESSES

Purpose

To specify the maximum number of gateway processes that an instance of Oracle Connection Manager supports.

Values

The number designated must be greater than the minimum number of gateway processes. The maximum is 64.

9.2.29 MAX_REG_CONNECTIONS

Purpose

To specify the maximum number of concurrent registration connection sessions that can be supported by Oracle Connection Manager.

Default

512

Example

```
MAX_REG_CONNECTIONS=20
```

9.2.30 MIN_GATEWAY_PROCESSES

Purpose

To specify the minimum number of gateway processes that an instance of Oracle Connection Manager supports.

Values

Any number of sessions can be designated up to 64.

9.2.31 NEXT_HOP

The `NEXT_HOP` parameter provides static routing of client connections from Oracle Connection Manager (Oracle CMAN).

Purpose

To specify a fixed address for Oracle CMAN to connect and to relay all client connection requests.

Usage Notes

This parameter contains the next hop address to which Oracle CMAN should connect to, whenever there is a client connection to it. This parameter provides static routing of client connections from Oracle CMAN and does not require service registration.

Values

You must specify this parameter in the `CONFIGURATION` section. Use `description` or `address list` to specify multiple addresses along with other characteristics such as `load_balance` and `failover`.

Default

Not enabled.

Example

```
CMAN=
(CONFIGURATION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=4555))
    (rule_list=(rule=(src=*)(dst=*)(srv=*)(act=accept)))
    (PARAMETER_LIST=
      (MAX_GATEWAY_PROCESSES=8)
      (MIN_GATEWAY_PRCESSSES=3))
    (NEXT_HOP=(ADDRESS=(PROTOCOL=tcps)(HOST=proxysvr1)(PORT=1555))
     )
```

9.2.32 OUTBOUND_CONNECT_TIMEOUT

Purpose

To specify the length of time in seconds that the Oracle Connection Manager instance waits for a valid connection to be established with the database server or with another Oracle Connection Manager instance.

Values

- `60` to disable the timeout. This is the default.
- Any number greater than 0 to enable the timeout. The number equals the timeout period in seconds.

9.2.33 REGISTRATION_EXCLUDED_NODES

The Oracle Connection Manager parameter file (`cman.ora`) REGISTRATION_EXCLUDED_NODES specifies the list of nodes that cannot register with the listener.

Purpose

To specify the list of nodes that cannot register with the listener.

Usage Notes

The list can include host names or CIDR notation for IPv4 and IPv6 addresses. The wildcard format (`*`) is supported for IPv4 addresses. The presence of a host name in the list results in the inclusion of all IP addresses mapped to the host name. The host name should be consistent with the public network interface.

If the `REGISTRATION_INVITED_NODES` parameter and the `REGISTRATION_EXCLUDED_NODES` parameter are set, then the `REGISTRATION_EXCLUDED_NODES` parameter is ignored.

Values

Valid nodes and subnet IP addresses or names.

Example

```
REGISTRATION_EXCLUDED_NODES = 10.1.26.*, 10.16.40.0/24, \
                                         2001:DB8:3eff:fe38, node2
```

9.2.34 REGISTRATION_INVITED_NODES

The Oracle Connection Manager parameter file (`cman.ora`) `REGISTRATION_EXCLUDED_NODES` parameter specifies the list of node that can register with the listener.

Purpose

To specify the list of node that can register with the listener.

Usage Notes

The list can include host names or CIDR notation for IPv4 and IPv6 addresses. The wildcard format (`*`) is supported for IPv4 addresses. The presence of a host name in the list results in the inclusion of all IP addresses mapped to the host name. The host name should be consistent with the public network interface.

If the `REGISTRATION_INVITED_NODES` parameter and the `REGISTRATION_EXCLUDED_NODES` parameter are set, then the `REGISTRATION_EXCLUDED_NODES` parameter is ignored.

Values

Valid nodes and subnet IP addresses or names.

Example

```
REGISTRATION_INVITED_NODES = 10.1.35.*, 10.1.34.0/24, \
                                        2001:DB8:fe38:7303, node1
```

9.2.35 REST_ADDRESS

Use the `REST_ADDRESS` parameter to configure REST endpoint hostname and port. Oracle CMAN listens to `tcps` endpoint based on the specified hostname and port.

Usage Notes

Add the `REST_ADDRESS` attribute under the `parameter_list` of the `cman.ora` file.

Syntax

```
REST_ADDRESS=host name:port
```

Example

```
REST_ADDRESS=cman_host:1524
```

9.2.36 RULE

Purpose

To specify an access control rule list to filter incoming connections.

Usage Notes

A rule list specifies which connections are accepted, rejected, or dropped.

If no rules are specified, then all connections are rejected.

The source and destination can be a host name, IP address, or subnet mask.

There must be at least one rule for client connections and one rule for CMCTL connections. Omitting one or the other results in the rejection of all connections for the rule type omitted. The last rule in the example that follows is a CMCTL rule.

Oracle Connection Manager does not support wildcards for partial IP addresses. If you use a wildcard, then use it in place of a full IP address. The IP address of the client may, for example, be (SRC=*).

Oracle Connection Manager supports only the `/nn` notation for subnet addresses. In the first rule in Example "**Sample cman.ora File**", /27 represents a subnet mask that comprises 27 left-most bits.

Values

This parameter is listed in the rule list section of the `cman.ora` file preceded by `RULE_LIST=`.

Syntax

```
(RULE_LIST=
  (RULE=
```

```
    (SRC=host)
    (DST=host)
    (SRV=service_name)
    (ACT={accept|reject|drop})
    (ACTION_LIST=AUT={on|off}
    ((CONN_STATS={yes|no})(MCT=time)(MIT=time)(MOCT=time)))
  (RULE= ...))
```

Additional Parameters

The `RULE` parameter filters a connection or group of connections using the following parameters:

`SRC`: The source host name or IP address of the client.

`DST`: The destination server host name or IP address of the database server.

`SRV`: The database service name of Oracle Database obtained from the `SERVICE_NAME` parameter in the initialization parameter file.

`ACT`: The action for the connection request. Use `accept` to accept incoming requests, `reject` to reject incoming requests, or `drop` to reject incoming requests without sending an error message.

`ACTION_LIST`: The rule-level parameter settings for some parameters. These parameters are as follows:

- `AUT`: Oracle Database security authentication on client side.
- `CONN_STATS`: Log input and output statistics.
- `MCT`: Maximum connect time.
- `MIT`: Maximum idle timeout.
- `MOCT`: Maximum outbound connect time.

Rule-level parameters override their global counterparts.

Example

```
(RULE_LIST=
  (RULE=
    (SRC=client1-pc)
    (DST=sales-server)
    (SRV=sales.us.example.com)
    (ACT=reject))
  (RULE=
    (SRC=192.0.2.45)
    (DST=192.0.2.200)
    (SRV=db1)
    (ACT=accept))
  (RULE=
    (SRC=sale-rep)
    (DST=sales1-server)
    (SRV=cmon)
    (ACT=accept)))
```

9.2.37 SDU

Purpose

To specify the session data unit (SDU) size, in bytes, to connections

Usage Notes

Oracle Connection Manager can negotiate large SDU with client and server when configured. When the configured values of client, database server, and Oracle Connection Manager do not match for a session, the least value of all the three values is used.

Default

8192 bytes (8 KB)

Values

512 to 2097152 bytes

Example

```
SDU=32768
```

9.2.38 SERVICE_RATE

The SERVICE_RATE parameter of cman.ora file specifies incoming connection rate that is allowed per service for an instance.

Purpose

To specify incoming connection rate that is allowed per service for an instance.

Usage Notes

Any user-specified value greater than `0` sets the maximum limit on the number of new connections per service-instance handled by the proxy listener every second. Listener rejects connections after it reaches the maximum limit. Client side connection failure is reported with “TNS:listener: rate limit reached”.

Values

- `0` to disable service rate limit. This is the default.
- Any number greater than 0 to enable service rate limit.

Example 9-6 Example

```
SERVICE_RATE=10
```

9.2.39 SESSION_TIMEOUT

Purpose

To specify the maximum time in seconds allowed for a user session.

Usage Notes

The global setting can be overridden by a rule-level setting in `ACTION_LIST`.

Values

- `0` to disable the timeout. This is the default.
- Any number greater than 0 to enable the timeout. The number equals the timeout period in seconds.

9.2.40 SSL_CIPHER_SUITES

Use the `SSL_CIPHER_SUITES` parameter to control the combination of authentication, encryption, and data integrity algorithms used by Transport Layer Security (TLS).

Purpose

To control the combination of authentication, encryption, and data integrity algorithms used by TLS. By default, the strongest protocol and cipher are negotiated between the database client and server. Setting this parameter will override the default behavior. You must use this parameter only if you have internal security controls that dictate the usage of certain protocol versions.

Usage Notes

Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.

Enclose the `SSL_CIPHER_SUITES` parameter value in parentheses. Otherwise, the cipher suite setting does not parse correctly.

Default

None

Values

Approved ciphers compatible with TLS 1.3:

- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256` (non-FIPS only)
- `TLS_AES_128_CCM_SHA256`
- `TLS_AES_128_GCM_SHA256`

Approved ciphers compatible with TLS 1.2:

- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

Deprecated ciphers compatible with TLS 1.2:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

Examples

```
SSL_CIPHER_SUITES=(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

```
SSL_CIPHER_SUITES=(TLS_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
```

Related Topics

- Set the TLS Cipher Suites on the Server
- Set the TLS Cipher Suites on the Client

9.2.41 SSL_CLIENT_AUTHENTICATION

Use the `SSL_CLIENT_AUTHENTICATION` parameter to specify whether the database client is authenticated using Transport Layer Security (TLS).

Purpose

To enable client authentication in a TLS connection. The connection can be one-way or two-way (mutual TLS or mTLS).

Usage Notes

When set to `TRUE`, a two-way TLS connection is initiated. Both the client and server (including the listener) authenticate each other. For example, if you set this parameter to `TRUE` in the server configuration (server-side `sqlnet.ora`), then the server attempts to authenticate the client. If you set it to `TRUE` in the listener configuration (`listener.ora`), then the listener attempts to authenticate the client.

When set to `FALSE`, only the client authenticates the server and listener as a one-way TLS connection. For example, if you set this parameter to `FALSE` in the server configuration, then the server does not authenticate the client. If you set it to `FALSE` in the listener configuration, then the listener does not authenticate the client.

When set to `OPTIONAL`, the server behaves as follows:

- If the client sends a certificate, then the connection is completed as a two-way TLS connection after authenticating the client.
- If the client does not send a certificate, then the connection is completed as a one-way TLS connection.

Ensure that this parameter setting is consistent for the server or listener (on one side) and the client (on the other). Otherwise, the connection may fail. For example, if you enable client authentication in the server or listener configuration, then you must enable it in the client configuration.

Default

`TRUE`

Values

- `TRUE` | `ON` | `YES` | `1`: To enable mTLS
- `FALSE` | `OFF` | `NO` | `0`: To enable one-way TLS
- `OPTIONAL`: To enable both TLS and mTLS

Example

```
SSL_CLIENT_AUTHENTICATION=FALSE
```

Related Topics

- *Oracle Database Security Guide*

9.2.42 SSL_VERSION

Use the `SSL_VERSION` parameter to define valid Transport Layer Security (TLS) versions to be used for connections.

Purpose

To define the version of TLS that must run on the systems with which the database server communicates. By default, the database server and client negotiate the strongest security protocol. Oracle does not recommend modifying this parameter, unless your security requirements mandate the usage of certain protocol versions.

Usage Notes

- Clients, listeners, and database servers must use compatible versions. Modify this parameter only when necessary to enforce the use of the more secure TLS protocol and not allow clients that only work with the older TLS protocols. The current default uses TLS 1.3, which is the version required for multiple security compliance requirements. If you need to specify TLS 1.2, then also include TLS 1.3 to allow more secure connections.
- In addition to `sqlnet.ora`, `listener.ora`, and `cman.ora`, you can specify this parameter under the `SECURITY` section of `tnsnames.ora` or directly as part of the connect string. The parameter value specified in the connect string takes precedence over the other specified values.
- Starting with Database 23ai, the use of Transport Layer Security protocol versions 1.0 and 1.1 are desupported.

  In most cases, this change will not have any impact, because the database client and server will negotiate the use of the most secure protocol and cipher algorithm. However, if TLS 1.0 or 1.1 has been specified, then you must either remove it to allow the database server and client to pick the most secure protocol, or you must specify either TLS 1.2, or TLS 1.3, or both, for the protocol. Oracle recommends using the latest, most secure protocol. That protocol is TLS 1.3, which is introduced with Oracle Database 23ai.
- Starting with Oracle Database 23ai, the Secure Socket Layer v3 protocol (SSLv3) is no longer supported for database server-client connections, and the `sqlnet.ora` parameter `ADD_SSLV3_TO_DEFAULT` has been removed.

  SSLv3 is a much less secure protocol to secure the database server-to-client connection. Instead of using SSLv3, allow the database server and client to negotiate the most secure protocol that is common between the server and the client. Oracle Database 23ai provides TLS 1.2 and TLS 1.3 protocols for certificate-based network encryption.
- If you set `SSL_VERSION` to `undetermined`, then the most secure TLS protocol version is used. You can also use the `SSL_VERSION=undetermined` setting in the connect string for a specific connection to override the `SSL_VERSION` value configured in the `sqlnet.ora`, `listener.ora`, or `cman.ora` file.
- If you do not set `SSL_VERSION` to any value, then all the supported TLS protocol versions are tried starting with the most secure version. This is typically the most common configuration, ensuring that the strongest protocol is chosen during TLS negotiation.

Values

`undetermined` | `TLSv1.2` | `TLSv1.3`

Default

`undetermined`

Syntax and Examples

- To specify a single protocol version:

  `SSL_VERSION=`*`TLS_protocol_version`*

  For example:

  `SSL_VERSION=TLSv1.3`

- To specify multiple protocol versions, use a comma-separated string of values, enclosed in parenthesis:

  `SSL_VERSION=(`*`TLS_protocol_version1`*`,`*`TLS_protocol_version2`*`)`

  For example:

  `SSL_VERSION=(TLSv1.2,TLSv1.3)`

> **Note:**
>
> Do not enclose protocol versions in parenthesis while specifying this parameter in the `tnsnames.ora` file or as part of the connect string, otherwise the setting will not parse correctly. For example:
>
> ```
> net_service_name=
>   (DESCRIPTION=
>      (ADDRESS=(PROTOCOL=tcps)(HOST=salesserver)(PORT=1522))
>      (SECURITY=(SSL_VERSION=TLSv1.2,TLSv1.3))
>   )
> ```

Related Topics

- Set the Required TLS Version on the Server
- Set the Required TLS Version on the Client

9.2.43 TRACE_FILE

Purpose

To specify the directory for Oracle Connection Manager trace files.

9.2.44 TRACE_FILELEN

Purpose

To specify the size of the trace file in KB.

Usage Notes

When the size is reached, the trace information is written to the next file. The number of files is specified with the `TRACE_FILENO` parameter.

9.2.45 TRACE_FILENO

Purpose

To specify the number of trace files.

Usage Notes

When this parameter is set along with the `TRACE_FILELEN` parameter, trace files are used in a cyclical fashion. The first file is filled first, then the second file, and so on. When the last file has been filled, the first file is reused, and so on.

9.2.46 TRACE_LEVEL

Purpose

To specify the level for trace messages.

Values

- `off` for no tracing. This is the default.
- `user` for user-induced errors trace information.
- `admin` for administration trace information, such as installation-specific.
- `support` for Oracle Support Services information.

9.2.47 TRACE_TIMESTAMP

Purpose

To specify the use of a timestamp for the tracing logs.

Usage Notes

If the `TRACING` parameter is enabled, then a time stamp in the form of `dd-mmm-yyyy hh:mi:ss:mil` for every trace event in the trace file.

Values

- `off` for no timestamp to be included in the file.
- `on` for timestamp to be included in the file.

9.2.48 USE_SERVICE_AS_TNSNAMES_ALIAS

Use this parameter for static routing of client connections from Oracle connection manager based on client's service name.

Usage Notes

Oracle connection manager uses the service name specified by the client as an alias. You must configure alias in `tnsnames.ora` file of CMAN home. If an alias is not configured for a service, then the `NEXT_HOP` parameter in `cman.ora` acts as a default connect string.

Values

`OFF` and `ON`. The default is `OFF`.

Example

Configuration in CMAN home:

```
cman.ora

USE_SERVICE_AS_TNSNAMES_ALIAS=ON


tnsnames.ora


sales=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(port=1521))
   (CONNECT_DATA=(SERVICE_NAMES=sales)))
```

> **Note:**
>
> `DESCRIPTION_LIST` is not supported in the `tnsnames.ora` file of CMAN home.

Configuration in client home:

```
tnsnames.ora


sales_cman=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=TCP)(HOST=cman-server)(port=1523))
   (CONNECT_DATA=(SERVICE_NAMES=sales)))
```

In this example, the client is connecting to service `sales`. CMAN will use the `sales` alias in `tnsnames.ora` of the CMAN home for connecting to the next hop.

9.2.49 USE_SID_AS_SERVICE

The `USE_SID_AS_SERVICE` Oracle Connection Manager parameter enables the system identifier (SID) in the connect descriptor to be interpreted as a service name when a user attempts a database connection.

Purpose

To enable the system identifier (SID) in the connect descriptor to be interpreted as a service name when a user attempts a database connection.

Usage Notes

Database clients with earlier releases of Oracle Database that have hard-coded connect descriptors can use this parameter to connect to a container or pluggable database.

For an Oracle container database, the client must specify a service name in order to connect to it. Setting this parameter to `on` instructs the Oracle Connection Manager listener to use the SID in the connect descriptor as a service name and connect the client to the specified database.

Values

- `off` (default value)
- `on`

Example 9-7 Example

```
USE_SID_AS_SERVICE=on
```

9.2.50 VALID_NODE_CHECKING_REGISTRATION

Purpose

To determine whether valid node checking registration is performed, and if the subnet is allowed.

Usage Notes

When set to `on`, valid node checking registration is performed at the listener for any incoming registration request, and only local IP addresses are allowed.

Default

`on`

Values

- `off | 0` to specify valid node checking registration is off, and no checking is performed.
- `on | 1 | local` to specify valid node checking registration is on, and all local IP addresses can register. If a list of invited nodes is set, then all IP addresses, host names, or subnets in the list as well as local IP addresses are allowed.

- `subnet | 2` to specify valid node checking registration is on, and all machines in the local subnets are allowed to register. If a list of invited nodes is set, then all nodes in the local subnets as well as all IP addresses, host names and subnets in the list are allowed.

Example

```
VALID_NODE_CHECKING_REGISTRATION = on
```

9.2.51 WALLET_LOCATION

Use the `WALLET_LOCATION` parameter to specify the location of Oracle wallets.

Purpose

To specify the directory path where you want to create and store an Oracle wallet. Wallets securely contain certificates, secrets, private keys, and trust points used by Oracle Database.

Usage Notes

- Deprecation of the server-side setting:

  The parameter `WALLET_LOCATION` is deprecated for use with Oracle Database 23ai for the Oracle Database server. It is not deprecated for use with the Oracle Database client.

  For Oracle Database server, Oracle recommends that you use the `WALLET_ROOT` system parameter instead of using `WALLET_LOCATION`.

- Where to set this parameter:

  You can set `WALLET_LOCATION` in the `sqlnet.ora` file to specify a common wallet location for all connections. You can also set it in the connect string or `tnsnames.ora` file to specify a different wallet location for a particular connection.

  Use of `WALLET_LOCATION` in the connect string or `tnsnames.ora` overrides the `sqlnet.ora` `WALLET_LOCATION` setting for the specific `tnsnames.ora` service. The `tnsnames.ora` `WALLET_LOCATION` setting enables a client to initiate multiple TLS sessions using different TLS certificates in the same client process.

- Setting to use the system default certificate store instead of a client-side wallet:

  The Linux and Windows database clients can use the system default certificate store to validate the Oracle Database server certificate, instead of creating a local wallet with root certificate. The default certificate store is located in `/etc/pki/tls/cert.pem` on Linux and Microsoft Certificate Store (MCS) on Windows.

  If you set `WALLET_LOCATION=SYSTEM` in the connect string (in `tnsnames.ora` or directly to the command line), then the database client uses the default certificate store to validate the server certificate. In this case, the server certificate needs to be signed by a trusted root certificate that is already installed in the default certificate store.

  For example:

```
net_service_name=
    (DESCRIPTION =
       (ADDRESS=(PROTOCOL=tcps)(HOST=sales-svr)(PORT=1234))
```

```
    (SECURITY=(WALLET_LOCATION=SYSTEM))
    (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com))
  )
```

- Order in which the database client searches for a client wallet:
  1. The database client first tries to use a wallet from the `WALLET_LOCATION` directory specified in the connect string.
  2. If no wallet is present, then the client searches for the `WALLET_LOCATION` parameter value in the `sqlnet.ora` file.
  3. If no wallet is present, then the client searches for a wallet in the `$TNS_ADMIN` environment variable directory.
  4. If no wallet is present, then the client searches in the default wallet location, that is, `/etc/ORACLE/WALLETS/`*`username`* on Linux and `C:\Users\`*`username`*`\ \ORACLE\WALLETS` on Windows.
  5. If no wallet is present, then the client uses the wallet from the system default certificate store.

  You can specify `WALLET_LOCATION` as `SYSTEM` in the connect string to ignore all the wallet configurations and use the system default certificate store instead.
- Setting for walletless TLS connections:
  The `WALLET_LOCATION` parameter is optional for TLS connections that do not use a client wallet. If you do not include `WALLET_LOCATION` in the connect string, `tnsnames.ora`, or `sqlnet.ora`, then the driver automatically picks up common root certificates from the system default certificate store (if the system is Windows or Linux).

  However, you may need to perform additional steps in the following cases:
  - If `WALLET_LOCATION` is set in `sqlnet.ora` for all connections, then you can override this setting for a specific connection that does not need a client wallet (using `WALLET_LOCATION=SYSTEM` in the connect string).
  - If a wallet is present in the `$TNS_ADMIN` environment variable directory, then the database client uses the `$TNS_ADMIN` path as the default wallet location. In this case, you can either override the `WALLET_LOCATION` setting (using `WALLET_LOCATION=SYSTEM` in the connect string) or remove that wallet.
- Storage of wallet files:
  The password-protected wallet is stored in an `ewallet.p12` file. The auto-login and local auto-login wallets are stored in a `cwallet.sso` file.

  For example, if an Oracle wallet is stored in the Microsoft Windows registry and the wallet's key `(KEY)` is `SALESAPP`, then the storage location of the password-protected wallet is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\EWALLET.P12`. The storage location of the auto-login and local auto-login wallets is `HKEY_CURRENT_USER\SOFTWARE\ORACLE\WALLETS\SALESAPP\CWALLET.SSO`.

Additional Parameters

Use `SOURCE` to specify the type of storage and storage location for wallets, as follows:

- `METHOD`: Type of storage
- `METHOD_DATA`: Storage location:
  - `DIRECTORY`: Location of wallet on the file system

- KEY: Wallet type and location in the Microsoft Windows registry

Syntax and Examples

The syntax depends on the wallet as follows:

- Wallet on the file system:

```
WALLET_LOCATION=
  (SOURCE=
    (METHOD=file)
    (METHOD_DATA=
       (DIRECTORY=directory)))
```

For example:

```
WALLET_LOCATION=
  (SOURCE=
      (METHOD=file)
      (METHOD_DATA=
         (DIRECTORY=/etc/oracle/wallets/databases)))
```

- Microsoft certificate store:

```
WALLET_LOCATION=
  (SOURCE=
     (METHOD=mcs))
```

The key-value pair for MCS omits the METHOD_DATA parameter because MCS does not use wallets. Instead, Oracle PKI (public key infrastructure) applications obtain certificates, trust points and private keys directly from a user's profile.

- Wallet in the Microsoft Windows registry:

```
WALLET_LOCATION=
   (SOURCE=
      (METHOD=reg)
      (METHOD_DATA=
         (KEY=registry_key)))
```

For example:

```
WALLET_LOCATION=
   (SOURCE=
     (METHOD=reg)
     (METHOD_DATA=
        (KEY=SALESAPP)))
```

Default

None

Related Topics

- *Oracle Database Security Guide*

9.3 Oracle Connection Manager in Traffic Director Mode Parameters

This section lists and describes the following cman.ora file parameters:

- SERVICE_AFFINITY
  Use the `cman.ora` parameter `SERVICE_AFFINITY` to modify the default load distribution mechanism for Oracle Connection Manager in Traffic Director Mode.
- TDM
- TDM_BIND_THREAD
- TDM_DATATYPE_CHECK
- TDM_PERPDB_PRCP_CONNFACTOR
  Use the `cman.ora` parameter `TDM_PERPDB_PRCP_CONNFACTOR` to configure per-PDB Proxy Resident Connection Pooling (PRCP).
- TDM_PRCP_MAX_CALL_WAIT_TIME
- TDM_PRCP_MAX_TXN_CALL_WAIT_TIME
- TDM_SHARED_THREADS_MAX
- TDM_SHARED_THREADS_MIN
- TDM_STATS_FREQUENCY
  Use the `cman.ora` parameter `TDM_STATS_FREQUENCY` to configure the frequency at which usage statistics are uploaded to PDB for per-PDB Proxy Resident Connection Pooling (PRCP) connections.
- TDM_THREADING_MODE

9.3.1 SERVICE_AFFINITY

Use the `cman.ora` parameter `SERVICE_AFFINITY` to modify the default load distribution mechanism for Oracle Connection Manager in Traffic Director Mode.

Purpose

To configure load distribution mechanism for Oracle Connection Manager in Traffic Director Mode. By default, Oracle Connection Manager in Traffic Director Mode uses service affinity to select a gateway for routing incoming connection requests. All new connection requests are routed to the gateways associated with database services.

Usage Notes

If you set this parameter to `ON`, then all new connection requests are routed to the gateways associated with database services.

If you set this parameter to `OFF`, then all new connection requests are routed to the least-loaded gateways.

When using Proxy Resident Connection Pooling (PRCP), Oracle recommends that you set the `SERVICE_AFFINITY` parameter to `OFF` for better performance and resource utilization of gateway processes.

Values

`ON` | `OFF`

Default

`ON`

Example

```
SERVICE_AFFINITY = {ON | OFF}
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Multitenant Administrator's Guide*

9.3.2 TDM

Purpose

To configure Oracle Connection Manager to act as Oracle Connection Manager in Traffic Director Mode.

Default

`FALSE`

Values

- `TRUE`
- `FALSE`

Example

```
tdm = TRUE
```

9.3.3 TDM_BIND_THREAD

Purpose

To make the application connection hold on to the TDM thread and has different implications with and without PRCP. This parameter only applies when `TDM_THREADING_MODE` is set to `SHARED`.

Usage Notes

Without PRCP, setting this parameter to `yes` makes the application connection hold on the TDM worker thread as long as there is a transaction in progress.

With PRCP, setting this parameter to `yes` makes the application connection hold on to the TDM thread from the time `OCISessionGet` is done by the application till it does an `OCISessionRelease`.

Default

`no`

Values

- `yes`

- no

Example

```
TDM_BIND_THREAD = yes
```

9.3.4 TDM_DATATYPE_CHECK

Purpose

To validate all the inbound data to the database, of the data type `NUMBER`, `DATE`, `TIMESTAMP`, `TIMESTAMP WITH LOCAL TIMEZONE`, `TIMESTAMP WITH TIMEZONE`, `BLOB`, `CLOB`, `BFILE`, `UROWID` and `REF`. The following error is received by the application if there is any problem with the data sent to the Oracle Connection Manager in Traffic Director Mode.

```
ORA-03137: malformed TTC packet from client rejected: [3101]
```

Usage Notes

Turning `ON/OFF` this parameter enables or disables the data validation.

Default

```
OFF
```

Values

- `ON`
- `OFF`

Example

```
tdm_datatype_check={ON | OFF}
```

9.3.5 TDM_PERPDB_PRCP_CONNFACTOR

Use the `cman.ora` parameter `TDM_PERPDB_PRCP_CONNFACTOR` to configure per-PDB Proxy Resident Connection Pooling (PRCP).

Purpose

To configure per-PDB PRCP. This parameter value sets a connection factor, which helps in dynamically determining the maximum size of every per-PDB PRCP pool.

Usage Notes

- The per-PDB PRCP setting determines the maximum size of a per-PDB PRCP pool based on the `TDM_PERPDB_PRCP_CONNFACTOR` parameter value and the Oracle Compute Unit (OCPU) count allocated to each PDB automatically.

  A background process automatically fetches these values and resizes the pool. This derived maximum size value overrides the `<session_pool> MAX_SIZE` parameter configured in the `oraaccess.xml` file.

- PRCP dynamically rereads the `TDM_PERPDB_PRCP_CONNFACTOR` value and accordingly refreshes the maximum size of a per-PDB pool, if needed. You can change this parameter value using the Oracle Connection Manager Control utility (CMCTL) `RELOAD`

command. There is no need to restart Oracle Connection Manager in Traffic Director Mode for the changes to take effect.

- In addition to `TDM_PERPDB_PRCP_CONNFACTOR`, you must set the `sqlnet.ora` parameter `TCP.ALLOWED_PROXIES` on the database server. Otherwise, the connection request fails. `TCP.ALLOWED_PROXIES` specifies the CMAN instance (IP address or host name) that can fetch the OCPU count from the database server.

Values

- `0` to disable per-PDB PRCP
- Any number equal to or greater than `1` to enable per-PDB PRCP

> **Note:**
>
> Ensure that you specify a connection factor value within the maximum connections limit defined by the `cman.ora` parameter `MAX_CONNECTIONS`.

Default

`0`

Example

```
TDM_PERPDB_PRCP_CONNFACTOR=10
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- TCP.ALLOWED_PROXIES
  Use the `sqlnet.ora` parameter `TCP.ALLOWED_PROXIES` to specify a list of the Oracle Connection Manager (CMAN) addresses that can forward client IP address to the database server.
- RELOAD
  Use the Oracle Connection Manager Control utility `RELOAD` command to make the utility dynamically reread parameters and rules.
- MAX_CONNECTIONS

9.3.6 TDM_PRCP_MAX_CALL_WAIT_TIME

Purpose

To record the maximum time of inactivity, in seconds, for a client after obtaining a session from the PRCP pool. This parameter is applicable when the Oracle Connection Manager in Traffic Director Mode is configured to have Proxy Resident Connection Pool.

Usage Notes

After obtaining a session from the PRCP pool, if the client application does not issue a database call for the time specified by `TDM_PRCP_MAX_CALL_WAIT_TIME` parameter, then the PRCP session is freed and the client connection is terminated. As a result, if the

client application attempts a round trip call on such a connection, then it receives an `ORA-3113` or `ORA-3115` error.

Default

30 seconds

Values

Any non negative value. However, Oracle recommends not to use a value of `0` as that implies that a connection can acquire a PRCP session for an indefinite amount of time

9.3.7 TDM_PRCP_MAX_TXN_CALL_WAIT_TIME

Purpose

To record the maximum time of inactivity, in seconds, for a client after it obtains a session from the Proxy Resident Connection Pool and starts a transaction. This parameter is applicable when the Oracle Connection Manager in Traffic Director Mode is configured to have PRCP.

Usage Notes

If the client application does not issue a database call for the time specified by `TDM_PRCP_MAX_TXN_CALL_WAIT_TIME` parameter while in a transaction, the PRCP session is freed, the transaction is rolled back, and the client connection is terminated. As a result, if the client application attempts a round trip call on such a connection, then it receives an `ORA-3113` or `ORA-3115` error.

Default

`0`

Values

Any nonnegative value. However, it is recommended not to use a value of`0` as it implies that a connection can acquire a PRCP session for an indefinite amount of time.

9.3.8 TDM_SHARED_THREADS_MAX

Purpose

To configure the maximum number of threads that an Oracle Connection Manager process in Traffic Director Mode should have, when `tdm_threading_mode` is set to `SHARED`.

Values

Any number can be designated for the maximum number of threads. For `DEDICATED` mode, the maximum number of threads is same as the maximum number of connections. In `SHARED` mode, though there is no fixed upper bound, it should ideally be proportional to the load.

9.3.9 TDM_SHARED_THREADS_MIN

Purpose

To configure the minimum number of threads that an Oracle Connection Manager process in Traffic Director Mode should have, when `tdm_threading_mode` is set to `SHARED`.

Values

Any number can be designated for the minimum number of threads. For `SHARED` mode, there is no limit enforced. However, the number of threads should be proportional to the load.

9.3.10 TDM_STATS_FREQUENCY

Use the `cman.ora` parameter `TDM_STATS_FREQUENCY` to configure the frequency at which usage statistics are uploaded to PDB for per-PDB Proxy Resident Connection Pooling (PRCP) connections.

Purpose

To specify the time interval, in minutes, at which usage statistics for Oracle Connection Manager in Traffic Director Mode should be uploaded to PDB if per-PDB PRCP is enabled.

These usage statistics help in monitoring the behavior of your connection pools. PDB administrators can query the dynamic database view `V$TDM_STATS` to view this statistical data.

Values

- `0` to disable statistics upload.
- Any number equal to or greater than `1` (up to the maximum value) to enable statistics upload. This value depends on your runtime load and connection pool usage.

Default Value

`0`

Minimum Value

`0`

Maximum Value

`2800`

Example

```
TDM_STATS_FREQUENCY=300
```

Related Topics

- *Oracle Database Net Services Administrator's Guide*
- V$TDM_STATS

9.3.11 TDM_THREADING_MODE

Purpose

To configure the usage of threads by the Oracle Connection Manager in Traffic Director Mode.

Usage Notes

If this parameter is set to `DEDICATED`, then a worker thread is spawned for each inbound connection and the maximum number of threads is determined by the `max_connections` parameter

If this parameter is set to `SHARED`, then a shared pool of worker threads handle all inbound connections. The minimum number of worker threads is specified by the `tdm_shared_threads_min` setting and the maximum number of worker threads is specified by the `tdm_shared_threads_max` setting. The thread pool is internally managed within these bounds.

Default

`DEDICATED`

Values

- `DEDICATED`
- `SHARED`

Example

```
tdm_threading_mode={DEDICATED | SHARED}

tdm_shared_threads_min = 4

tdm_shared_threads_max = 5
```

9.4 ADR Diagnostic Parameters for Oracle Connection Manager

The diagnostic data for critical errors is quickly captured and stored in the ADR for Oracle Connection Manager.

Since Oracle Database 11g, Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems. The problems are critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

When a critical error occurs, it is assigned an incident number, and diagnostic data for the error, such as traces and dumps, are immediately captured and tagged with the incident number. The data is then stored in the Automatic Diagnostic Repository (ADR), a file-based repository outside the database.

This section describes the parameters used when ADR is enabled. ADR is enabled by default. Non-ADR parameters listed in the `cman.ora` file are ignored when ADR is enabled.

- ADR_BASE
  It is a diagnostic parameter in the `cman.ora` file and it specifies the base directory to store tracing and logging incidents when ADR is enabled.
- DIAG_ADR_ENABLED
  `DIAG_ADR_ENABLED` diagnostic parameter of the `cman.ora` file indicates whether ADR tracing is enabled.
- LOG_LEVEL
- TRACE_LEVEL
- TRACE_TIMESTAMP

9.4.1 ADR_BASE

It is a diagnostic parameter in the `cman.ora` file and it specifies the base directory to store tracing and logging incidents when ADR is enabled.

Purpose

To specify the base directory to store tracing and logging incidents when ADR is enabled.

Default

The default is `ORACLE_BASE`, or `ORACLE_HOME/log` if `ORACLE_BASE` is not defined.

Values

Any valid directory path to a directory with write permission.

Example 9-8 Example

```
ADR_BASE=/oracle/network/trace
```

9.4.2 DIAG_ADR_ENABLED

`DIAG_ADR_ENABLED` diagnostic parameter of the `cman.ora` file indicates whether ADR tracing is enabled.

Purpose

To indicate whether ADR tracing is enabled.

Usage Notes

When the `DIAG_ADR_ENABLED` parameter is set to `OFF`, then non-ADR file tracing is used.

Values

`on` | `off`

Example 9-9 Example

```
DIAG_ADR_ENABLED=on
```

9.4.3 LOG_LEVEL

Purpose

To specify the level of logging performed by Oracle Connection Manager.

Usage Notes

This parameter is also applicable when non-ADR logging is used.

The following log files are used with Oracle Connection Manager:

- *instance-name_pid*.log for the listener.
- *instance-name*_cmadmin_*pid*.log for CMADMIN.
- *instance-name*_cmgw_*pid*.log for the gateway processes.

The log files are located in the ORACLE_HOME/network/log directory.

Default

off or 0

Values

- off or 0 for no log output.
- user or 4 for user log information.
- admin or 10 for administration log information.
- support or 16 for Oracle Support Services log information.

Example

```
LOG_LEVEL=admin
```

9.4.4 TRACE_LEVEL

Purpose

To specify the trace level for the Oracle Connection Manager instance.

Usage Notes

This parameter is also applicable when non-ADR tracing is used.

The following trace files are used with Oracle Connection Manager:

- *instance-name_pid*.trc for the listener.
- *instance-name*_cmadmin_*pid*.trc for CMADMIN.
- *instance-name*_cmgw_*pid*.trc for the gateway processes.

The log files are located in the ORACLE_HOME/network/log directory.

Default

off

Values

- `off` for no trace output.
- `user` for user trace information.
- `admin` for administration trace information.
- `support` for Oracle Support Services trace information.

Example

```
TRACE_LEVEL=admin
```

9.4.5 TRACE_TIMESTAMP

Purpose

To add a time stamp in the form of `dd-mmm-yyyy hh:mi:ss:mil` to every trace event in the trace file for the listener.

Usage Notes

This parameter is used with the TRACE_LEVEL parameter. This parameter is also applicable when non-ADR tracing is used.

Default

on

Values

- `on` or `true`
- `off` or `false`

Example

```
TRACE_TIMESTAMP=true
```

9.5 Non-ADR Diagnostic Parameters for Oracle Connection Manager

This section lists the parameters used when ADR is disabled:

- LOG_DIRECTORY
- TRACE_DIRECTORY
- TRACE_FILELEN
- TRACE_FILENO

9.5.1 LOG_DIRECTORY

Purpose

To specify the location of Oracle Connection Manager log files.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_BASE_HOME/network/log`

Values

Any valid directory path to a directory with write permission.

Example

```
LOG_DIRECTORY=/oracle/network/log
```

9.5.2 TRACE_DIRECTORY

Purpose

To specify the location of the Oracle Connection Manager trace files.

Usage Notes

Use this parameter when ADR is not enabled.

Default

`ORACLE_BASE_HOME/network/trace`

Values

Any valid directory path to a directory with write permission.

Example

```
TRACE_DIRECTORY=/oracle/network/admin/trace
```

9.5.3 TRACE_FILELEN

Purpose

To specify the size, in KB, of the trace file.

Usage Notes

When the size is met, the trace information is written to the next file. The number of files is specified with the TRACE_FILENO parameter. Any size can be designated. Use this parameter when ADR is not enabled.

Default

Unlimited

Example

```
TRACE_FILELEN=100
```

9.5.4 TRACE_FILENO

Purpose

To specify the number of trace files for Oracle Connection Manager tracing.

Usage Notes

When this parameter is set along with the TRACE_FILELEN parameter, trace files are used in a cyclical fashion. The first file is filled first, then the second file, and so on. When the last file has been filled, the first file is reused, and so on. Any number of files can be designated.

The trace file names are distinguished from one another by their sequence number. For example, if this parameter is set to `3`, then the gateway trace files would be named *instance-name*`_cmgw1_`*pid*`.trc`, *instance_name*`_cmgw2_`*pid*`.trc` and *instance_name*`_cmgw3_`*pid*`.trc`.

In addition, trace events in the trace files are preceded by the sequence number of the file. Use this parameter when ADR is not enabled.

Default

1

Example

```
TRACE_FILENO=3
```

9.6 Oracle Connection Manager Tunneling Parameters

This section lists the parameters that you must configure to enable tunneling.

- TUNNELING
  Set this parameter to start Oracle Connection Manager as server in tunneling mode.
- TUNNEL_CAPACITY
  Use this parameter to specify the number of reverse connections that can be multiplexed over a tunnel.
- MAX_TUNNELS
  Use this parameter to specify the number of tunnels that a client connection manager in tunneling mode can create.
- TUNNEL_PROBE_INTERVAL
  Use this parameter in server connection manager to keep the tunnel connection open.

- NON_TUNNEL_GATEWAYS
  Use this parameter to specify the number of regular gateways that will not be used for tunneling.
- TUNNEL_ADDRESS
  Set this parameter on the client CMAN to point to the server CMAN that you want to connect to.
- GATEWAY_PROCESSES
  Use this parameter to specify the number of gateway processes.

9.6.1 TUNNELING

Set this parameter to start Oracle Connection Manager as server in tunneling mode.

Purpose

Set this parameter to `ON` to start Oracle Connection Manager in tunneling mode. You must set this parameter on the server CMAN. When this parameter is set, the CMAN starts processing and accepts tunnel requests.

Usage Notes

Use this parameter with `PARAMETER_LIST`.

Default

`OFF`

Example

```
(PARAMETER_LIST=
      (TUNNELING=ON))
```

9.6.2 TUNNEL_CAPACITY

Use this parameter to specify the number of reverse connections that can be multiplexed over a tunnel.

Purpose

You must set this parameter on the server CMAN. Only the number of connections that you specify for this parameter will be allowed per tunnel.

Usage Notes

Use this parameter with `PARAMETER_LIST`.

Example

```
(PARAMETER_LIST=
```

```
      (TUNNELING_CAPACITY=25))
```

9.6.3 MAX_TUNNELS

Use this parameter to specify the number of tunnels that a client connection manager in tunneling mode can create.

Purpose

This parameter creates the specified number of tunnels by each connection manager gateway. You must set this parameter on the client CMAN.

Usage Notes

Use this parameter with `PARAMETER_LIST`.

Example

```
(PARAMETER_LIST=
      (MAX_TUNNELS=4))
```

9.6.4 TUNNEL_PROBE_INTERVAL

Use this parameter in server connection manager to keep the tunnel connection open.

Purpose

Specify a time interval in minutes to send small probe packets to keep the tunnel connection open and avoid time out. You must set this parameter on the server CMAN.

Usage Notes

Use this parameter with `PARAMETER_LIST`.

Example

```
(PARAMETER_LIST=
      (TUNNEL_PROBE_INTERVAL=7))
```

9.6.5 NON_TUNNEL_GATEWAYS

Use this parameter to specify the number of regular gateways that will not be used for tunneling.

Purpose

Set this parameter at both the server CMAN and the client CMAN to specify the number of regular gateways. Regular gateways handle regular and forward connections. In tunneling mode, all gateways are tunnel gateways by default.

Usage Notes

Use this parameter with `PARAMETER_LIST`.

Default

`0` when tunneling is enabled.

Example

```
(PARAMETER_LIST=
  (NON_TUNNEL_GATEWAYS=2))
```

9.6.6 TUNNEL_ADDRESS

Set this parameter on the client CMAN to point to the server CMAN that you want to connect to.

Purpose

The gateways connect to the specified server address to create tunnels. You can configure single or multiple addresses using address_list and description.

Usage Notes

Put this parameter under `CONFIGURATION`.

Example

```
(CONFIGURATION=
  (TUNNEL_ADDRESS=
  (DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=host_name)(PORT=port_number))
  (CONNECT_DATA=(TUNNEL_ID=tunnel_id)))))
```

9.6.7 GATEWAY_PROCESSES

Use this parameter to specify the number of gateway processes.

Usage

Use this parameter with `PARAMETER_LIST`.

gateway_processes=*value*

Example

```
(PARAMETER_LIST=
           (gateway_processes=8))
```

> **Note:**
>
> `MIN_GATEWAY_PROCESSES` parameter and `MAX_GATEWAY_PROCESSES` parameter are not supported with tunneling option.

10

Directory Usage Parameters in the ldap.ora File

This chapter provides a complete listing of the `ldap.ora` file configuration parameters.

10.1 Overview of Directory Server Usage File

The `ldap.ora` file contains directory usage configuration parameters created by Oracle Internet Directory Configuration Assistant, or Oracle Net Configuration Assistant. Do not modify these parameters or their settings.

When created with Oracle Internet Directory Configuration Assistant, `ldap.ora` is located in the *`ORACLE_HOME`*`/ldap/admin` directory. When created with Oracle Net Configuration Assistant, the `ldap.ora` file is located either in the *`ORACLE_BASE_HOME`*`/network/admin` directory or the *`ORACLE_HOME`*`/network/admin` directory. The `ldap.ora` file can also be stored in the directory specified by the `LDAP_ADMIN` or `TNS_ADMIN` environment variable.

Related Topics

- Oracle Internet Directory
- Oracle Net Configuration Assistant

10.2 Directory Usage Parameters

This section lists and describes the following `ldap.ora` file configuration parameters.

10.2.1 DEFAULT_ADMIN_CONTEXT

`DEFAULT_ADMIN_CONTEXT` `ldap.ora` file configuration parameter specifies the default directory for the creation, modification, or search of the connect identifiers.

Purpose

To specify the default directory entry that contains an Oracle Context from which connect identifiers can be created, modified, or looked up.

Values

Valid distinguished name (DN)

Example 10-1 Example

```
DEFAULT_ADMIN_CONTEXT="o=OracleSoftware,c=US"
```

10.2.2 DIRECTORY_SERVER_TYPE

`DIRECTORY_SERVER_TYPE` is a networking parameter of the`ldap.ora` file and it specifies the type of directory server that is being used.

Purpose

To specify the type of directory server that is being used.

Values

- `oid` for Oracle Internet Directory
- `ad` for Microsoft Active Directory

Example 10-2 Example

```
DIRECTORY_SERVER_TYPE=oid
```

10.2.3 DIRECTORY_SERVERS

`DIRECTORY_SERVERS` is a directory usage parameter and it lists the host names and port number of the primary and alternate LDAP directory servers.

Purpose

To list the host names and port number of the primary and alternate LDAP directory servers.

Values

host:port[:sslport]

Example 10-3 Example

```
DIRECTORY_SERVERS=(ldap-server1:389:636, ldap-server2:389:636)
```

Appendices

Review information about features no longer supported in this release, upgrade concerns, and information about the Oracle Net Services LDAP schema.

- Upgrade Considerations for Oracle Net Services
  This appendix describes the coexistence and upgrade issues for Oracle Net Services.
- LDAP Schema for Oracle Net Services
  This appendix describes the Oracle schema object classes and attributes defined in the directory server for Oracle Net Services objects. It does not describe object classes and attributes reserved for future functionality or used by other Oracle products.

A

Upgrade Considerations for Oracle Net Services

This appendix describes the coexistence and upgrade issues for Oracle Net Services.

A.1 Anonymous Access to Oracle Internet Directory

Typical users of directory naming (LDAP) require anonymous access to the Oracle Internet Directory for name lookup.

Oracle Internet Directory Setting

If you upgrade your Oracle Internet Directory software release 11*g* or later, then the default setting for Oracle Internet Directory changes to disallow anonymous access to the directory. The directory administrator must configure the directory to enable anonymous binds after upgrading the directory to release 11*g*. In addition, the way anonymous binds are configured in Oracle Internet Directory changed between Oracle Database 10*g* and Oracle Database 11*g*.

B

LDAP Schema for Oracle Net Services

This appendix describes the Oracle schema object classes and attributes defined in the directory server for Oracle Net Services objects. It does not describe object classes and attributes reserved for future functionality or used by other Oracle products.

- Structural Object Classes
  The Oracle schema supports the structural object classes for Oracle Net directory naming lookups.
- Attributes
  It lists the attributes used for the object classes. This list is subject to change.

B.1 Structural Object Classes

The Oracle schema supports the structural object classes for Oracle Net directory naming lookups.

Table B-1 Oracle Net Structural Object Classes

Object Class	Attributes	Description
orclDBServer	• orclNetDescName • orclVersion	Defines the attributes for database service entries.
orclNetAddress	• orclNetAddressString • orclNetProtocol • orclVersion	Specifies a listener protocol address.
orclNetAddressAux1	• orclNetHostname	Specifies an auxiliary object class to add attributes to an orclNetAddress entry.
orclNetAddressList	• orclNetAddrList • orclNetFailover • orclNetLoadBalance • orclNetSourceRoute • orclVersion	Specifies a list of protocol addresses.
orclNetDescription	• orclNetAddrList • orclNetInstanceName • orclNetConnParamList • orclNetFailover • orclNetLoadBalance • orclNetSdu • orclNetServiceName • orclNetSourceRoute • orclSid • orclVersion	Specifies a connect descriptor containing the protocol address of the listener and the connect information to the service.

Table B-1 (Cont.) Oracle Net Structural Object Classes

Object Class	Attributes	Description
orclNetDescriptionAux1	• orclNetSendBufSize • orclNetReceiveBufSize • orclNetFailoverModeString • orclNetInstanceRole	Specifies auxiliary object class to add attributes to an orclNetDescription entry.
orclNetDescriptionList	• orclNetDescList • orclVersion	Specifies a list of connect descriptors.
orclNetService	• orclNetDescName • orclVersion	Defines the attributes for network service name entries.
orclNetServiceAlias	• orclNetDescName • orclVersion	Defines the attributes for network service alias entries.

B.2 Attributes

It lists the attributes used for the object classes. This list is subject to change.

Table B-2 LDAP Schema Attributes for Oracle Net Services

Attribute	Description
`orclCommonContextMap`	Allows the mapping of more than one default Oracle Context in the directory server.
`orclNetAddrList`	Identifies one or more listener protocol addresses.
`orclNetAddressString`	Defines a listener protocol address.
`orclNetConnParamList`	Placeholder for connect data parameters.
`orclNetDescList`	Identifies one or more connect descriptors.
`orclNetDescName`	Identifies a connect descriptor or a list of connect descriptors.
`orclNetFailover`	Turns connect-time failover on for a protocol address list.
`orclNetFailoverModeString`	Instructs Oracle Net to fail over to a different listener if the first listener fails during runtime. Depending on the configuration, session or any `SELECT` statements that were in progress are automatically failed over.
`orclNetHostname`	Specifies the host name.
`orclNetInstanceName`	Specifies the instance name to access.
`orclNetInstanceRole`	Specifies a connection to the primary or secondary instance of an Oracle Real Application Clusters (Oracle RAC) configuration.
`orclNetLoadBalance`	Turns client load balancing on for a protocol address list.
`orclNetProtocol`	Identifies the protocol used in the `orclAddressString` attribute.

Table B-2 (Cont.) LDAP Schema Attributes for Oracle Net Services

Attribute	Description
orclNetReceiveBufSize	Specifies the buffer space limit for receive operations of sessions.
orclNetSdu	Specifies the session data unit (SDU) size.
orclNetSendBufSize	Specifies the buffer space limit for send operations of sessions.
orclNetServiceName	Specifies the database service name in the CONNECT_DATA portion.
orclNetSourceRoute	Instructs Oracle Net to use each address in order until the destination is reached.
orclSid	Specifies the Oracle system identifier (SID) in the CONNECT_DATA portion of a connection descriptor.
orclVersion	Specifies the version of software used to create the entry.