Protocol Switching with Oracle Connection Manager

AN ORACLE WHITE PAPER / DECEMBER 5, 2019

ORACLE

PURPOSE STATEMENT

Oracle Connection Manager (CMAN) provides different methods to enable protocol switching, thereby allowing client-server connectivity where both sides use different protocols for transmission. This paper provides an overview of different approaches used by CMAN for routing Oracle Net traffic through different protocols, and it gives use cases, examples, and configuration details of the `NEXT_HOP` feature in CMAN.

DISCLAIMER

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

TABLE OF CONTENTS

INTRODUCTION

Oracle Connection Manager (CMAN) is a transparent proxy through which a client connection request is routed to the next hop. The next hop can be another CMAN, a SCAN Listener, or the database Listener. CMAN hides the database subnet, including any RAC cluster, from clients. Multiple databases can be deployed behind CMAN, while clients are provided a single point of entry into the database. Clients can transparently take advantage of features configured in CMAN, for example, session multiplexing, compression, TLS security, and access control.

CMAN also performs protocol switching while handling incoming connections. Protocol switching is required when incoming and outgoing connections use different protocols. For example, CMAN can receive incoming client connections over TCPS (TCP with TLS) and use TCP for outgoing connections, or vice versa. Protocol switching done by CMAN is transparent to both client and server, in other words, no changes are needed in the application in order to use this feature.

There are three ways in which CMAN can proxy incoming connections to the next hop network address:

- Dynamic Service registration: The database registers its services and protocol addresses with CMAN dynamically using the `remote_listener` database parameter. When CMAN receives a client request for a service, it forwards that connection to the appropriate service handler based on the registered services information.
- SOURCE_ROUTE: The client can specify `source_route` and provide the next hop protocol address in a connect string when connecting to CMAN. CMAN uses this address to forward this connection.
- NEXT_HOP: A fixed next hop address can be configured in CMAN's configuration. CMAN uses this address to forward all incoming connections.

USE CASES

Protocol switching is useful in various deployment scenarios, such as the following:

1. Older clients that do not support latest TLS versions. These clients can route connections to the database server via a CMAN instance, thereby leveraging newer TLS versions supported by CMAN. Here, CMAN performs protocol switching between TCP and TCPS (TLS over TCP) in order to enable communication between client and server.
2. Protocol switching is also useful in deployments where one side is inside a secure network and the other is in a non-secure network. This provides flexibility to use a non-secure protocol by either client or server, thereby offloading encryption and decryption overheads to CMAN.
3. With the use of `next_hop` in CMAN, there is no need for remote service registration by the database. It is also useful in deployments where a single outbound point is needed for different clients.
4. CMAN can be used as a bridge between IPv4 and IPv6 networks.

USING NEXT_HOP FOR PROTOCOL SWITCHING

One of the methods for protocol switching and forwarding client requests is by configuring a fixed next hop protocol address in CMAN. `NEXT_HOP` support is available starting with version 18c.

An Example

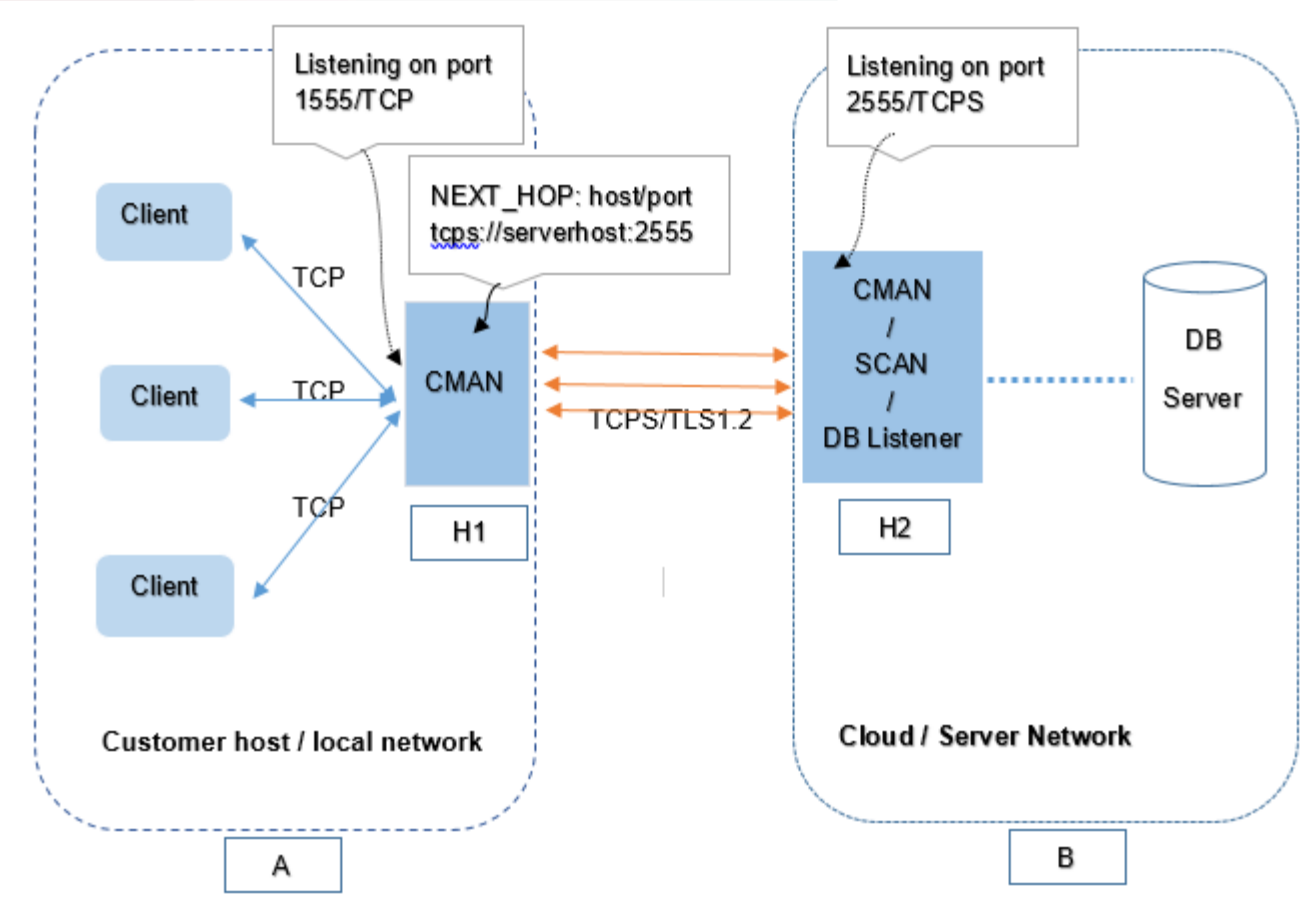

Figure 1: CMAN deployment for clients using TCP to TCPS protocol conversion for connecting to a database

In the deployment shown in Figure 1, clients are shown in Box A on the left, and servers are in Box B. Clients are using an older Oracle version where libraries do not support the latest TLS version (1.2). CMAN (version 18c or above) is deployed in Box A, which represents either an application server host or a client subnet. The exact placement of CMAN depends on security requirements. This CMAN listens on TCP address 1555 and has `next_hop` configured to server side listening endpoint, which is port 2555/TCPS. This CMAN supports TLS 1.2.

In this example, if H2 is SCAN Listener or database listener, the connection between client and server would be as follows:

Client **--[TCP]-->** CMAN **--[TCPS]-->** Database

H2 can also be another CMAN, in which case, the connection between client and server could possibly be as follows:

Client **--[TCP]-->** CMAN(H1) **--[TCPS]-->** CMAN(H2)**--TCP-->** Database

where both the CMANs are switching protocols between TCP and TCPS.

Protocol switching is not restricted to TCP and TCPS, or the above examples. Any supported SQL*Net protocol can be used.

CONFIGURING NEXT_HOP IN CMAN.ORA

The `NEXT_HOP` parameter is specified in a separate section under configuration in cman.ora.

A sample cman.ora with `next_hop` configured is:

```
CMAN=
     (CONFIGURATION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1555))
        (rule_list=
           (rule=(src=*)(dst=*)(srv=*)(act=accept))
         )
        (PARAMETER_LIST=
           (MAX_GATEWAY_PROCESSES=8)
           (MIN_GATEWAY_PRCESSSES=3)
         )
        (NEXT_HOP=(ADDRESS=(PROTOCOL=tcps)(HOST=serverhost)(PORT=2555)))
      )
WALLET_LOCATION= (source=(method=file)(method_data= (directory=$WALLET_LOC_PATH)))
```

NOTE: In the above example, `localhost` can be used instead of `proxysvr` in order to restrict incoming connections to the local node. This ensures that remote clients cannot connect into this CMAN in order to get routed to the database server, thereby ensuring that unencrypted protocols do not go over the wire even within the client network.

The value of the `next_hop` parameter is the protocol address of the next server which CMAN should connect to whenever a client connection request is received.

The possible format and values of `next_hop` are as follows:

a) Single address

```
(NEXT_HOP=(ADDRESS=(PROTOCOL=tcps)(HOST=serverhost)(PORT=2555)))
```

b) Multiple addresses: more than one address can be specified along with other characteristics, such as, `load_balance` and `failover` using `ADDRESS_LIST` or `DESCRIPTION`.

```
(NEXT_HOP=(ADDRESS_LIST=
                (LOAD_BALANCE=on)
```

```
                    (FAILOVER =on)
                    (ADDRESS=(PROTOCOL=tcps)(HOST=serverhost1)(PORT=2555))
                    (ADDRESS=(PROTOCOL=tcp)(HOST=serverhost1)(PORT=2556)))
```

or

```
    (NEXT_HOP=(DESCRIPTION=
                    (LOAD_BALANCE=on)
                    (FAILOVER=on)
                    (ADDRESS=(PROTOCOL=tcps)(HOST=serverhost1)(PORT=2555))
                    (ADDRESS=(PROTOCOL=tcps)(HOST=serverhost1)(PORT=2557)))
```

NOTE:

- Failover is on by default if multiple addresses are specified.
- When TCPS (TCP with TLS) protocol is used for the next hop protocol address, `wallet_location` needs to be specified in cman.ora, for example:

```
WALLET_LOCATION= (source=(method=file)(method_data= (directory=$WALLET_LOC_PATH)))
```

- `NEXT_HOP` is a reloadable parameter: after changing its value in cman.ora a restart of CMAN is not required. Reloading of parameters can be done using `cmctl` **reload** command.

VERIFICATION OF ESTABLISHED CLIENT CONNECTIONS VIA CMAN

On Linux, the `netstat` command can be used to verify that a connection has been established via CMAN. For example, when using SQL*Plus to connect to CMAN listening on local machine on port 1522:

```
$ netstat -anp | grep sqlplus
tcp        0      0 10.90.137.2:9992        10.90.137.2:1522        ESTABLISHED 13015/sqlplus
```

Next, processes using the client port 9992 can be identified:

```
$ netstat -anp | grep 9992
tcp        0      0 10.90.137.2:9992        10.90.137.2:1522        ESTABLISHED 13015/sqlplus
tcp6       0      0 10.90.137.2:1522        10.90.137.2:9992        ESTABLISHED 11796/cmgw
```

Note: The second line is for the peer, which is Gateway process (cmgw) in CMAN.

Next, connections for this gateway process can be viewed:

```
$ netstat -anp | grep 11796
tcp        0      0 10.90.137.2:43305       10.90.137.2:1521        ESTABLISHED 11796/cmgw
tcp6       0      0 10.90.137.2:1522        10.90.137.2:9992        ESTABLISHED 11796/cmgw
```

Note: The first line indicates that this gateway process is connected to DB Listener port 1521.

Additionally, tailing CMAN and Listener log files during connection establishment displays a new log entry in both files. During post-analysis, log entries can be matched using timestamps. For example, in the case of the above SQL*Plus connection, something like the following can be expected in the CMAN log:

```
26-SEP-2019 13:03:52 *
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=sqlplus)(HOST=myhost)(USER=self))) *
(ADDRESS=(PROTOCOL=tcp)(HOST=10.90.137.2)(PORT=9992)) * establish * sales.us.example.com * 0
```

and similarly, in Listener log:

```
26-SEP-2019 13:03:52 *
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=sqlplus)(HOST=myhost)(USER=self))) *
(ADDRESS=(PROTOCOL=tcp)(HOST=10.90.137.2)(PORT=43305)) * establish * sales.us.example.com * 0
```

Notice that the source port in the CMAN log entry (9992) is the same as shown by `netstat` for SQL*Plus, and the source port in the Listener log entry (43305) is the same as shown for the CMAN Gateway process.

The number of established connections can be monitored by using the `cmctl show connections` command. In order to use this command, CMAN must be configured with the parameter `(connection_statistics=yes)`. This command shows either a count of connections, or more details for each connection, for example:

```
Connection ID                 512
Gateway ID                    1
Source                        10.90.137.2
Destination                   10.90.137.2
Service                       sales.us.example.com
State                         ESTABLISHED
Idle time                     15
Connected time                15
Bytes Received [IN]           2783
Bytes Received [OUT]          4277
Bytes Sent     [IN]           4277
Bytes Sent     [OUT]          2783
```

CONCLUSION

The protocol switching feature of CMAN can be used in diverse deployment scenarios, including on-premise and Cloud. `NEXT_HOP` configuration in CMAN provides a way to route all incoming requests to a specific destination. This is very simple to setup with no registration complexity. It allows compatible older clients to communicate with newer servers using secure protocols which are not supported by older clients.

ORACLE CORPORATION

Worldwide Headquarters
500 Oracle Parkway, Redwood Shores, CA 94065 USA

Worldwide Inquiries
TELE + 1.650.506.7000 + 1.800.ORACLE1
FAX + 1.650.506.7200
oracle.com

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com. Outside North America, find your local office at oracle.com/contact.

 blogs.oracle.com/oracle  facebook.com/oracle  twitter.com/oracle

Integrated Cloud Applications & Platform Services

White Paper **Protocol Switching with Oracle Connection Manager**Protocol Switching with Oracle Connection ManagerProtocol Switching With Oracle Connection Manager
December 2019

 Oracle is committed to developing practices and products that help protect the environment