Chapter 23. Connection management through the Connection Manager

Connection Managers can control automatic failover for high-availability clusters, monitor client connections and direct requests to appropriate database servers, act as proxy servers and handle client/server communication, and prioritize connections between application servers and the primary server of a high-availability cluster. Connection Managers support high-availability clusters, replicate sets, server sets, and grids.

Automatic failover for database servers

If the Connection Manager detects that a primary server of a high-availability cluster has failed, it can promote a secondary server to the role of the primary server.

If you use multiple Connection Managers to manager failover for a cluster, you can enforce a consistent failover policy by setting the `onconfig` file HA_FOC_ORDER configuration parameter on the cluster's primary server. The value of the `onconfig` file HA_FOC_ORDER configuration parameter replaces the value of the `FOC` parameter's `ORDER` attribute in the configuration file of each Connection Manager that connects to the primary server.

Rule-based connection redirection and load balancing

Client applications can connect to a Connection Manager as if they are connecting to a database server. The Connection Manager gathers workload statistics from each server in the connection unit and uses service level agreements (SLAs) to manage and direct client connection requests to appropriate servers. When a client application makes a connection request through a redirect-mode SLA, the Connection Manager returns a database server's IP address and port number to the client application. The client application then uses the information to connect to the specified database server.

Connection Manager redirection takes place in the communication layer, so additional action is not required by client applications. Connection Managers can use redirection policies that are based on workload, latency, apply failures, or the apply backlog. Redirection can also be configured to occur round-robin.

Redirection-policy SLAs do not support connections from application that are compiled with Data Server Driver for JDBC and SQLJ version 3.5.1 or before, or with IBM Informix Client Software Development Kit (Client SDK) 3.00 or before.

Proxy-server connection management

Proxy-mode SLAs and redirect-mode SLAs are similar; in both cases, the Connection Manager gathers workload statistics from connection-unit servers, and controls which servers receive client connection request servers. When a client application makes a connection request through a proxy-mode SLA, client/server communication travels through the Connection Manager. When a database server is behind a firewall, Connection Managers can act as proxy servers, and handle client/server communication.

Proxy-policy SLAs do not have the same version restrictions that redirect-policy SLAs have. Connections from application that are compiled with any version of Data Server Driver for JDBC and SQLJ, or with any version of IBM Informix Client Software Development Kit (Client SDK) are supported.

Failover prioritization for application servers

You can install Connection Managers on the same hosts as application servers, and then prioritize the connections between each application server and the primary server of a high-availability cluster. This can help the highest priority application server maintain a connection to the cluster's primary server if a portion of the network fails.

Related concepts:

"Redirection and connectivity for data-replication clients" on page 24-6

"Failover configuration for high-availability clusters" on page 24-1

"Overview of DRDA" on page 2-43

"Components supporting high availability and scalability" on page 20-1

Configuring connection management

To configure connection management, you must install software, set environments and connectivity information, create Connection Manager configuration files, and run the **oncmsm** utility.

To configure and start connection management, complete the following steps:

1. Install at least one Connection Manager as part of the IBM Informix Client Software Development Kit (Client SDK) installation.
   a. If Connection Managers are installed on hosts where database servers are not installed, set each Connection Manager's host **INFORMIXDIR** environment variable to the directory the Connection Manager is installed into.
2. Modify connectivity information in the `sqlhosts` files that are on all client, database server, and Connection Manager hosts.
   a. If `sqlhosts` files are in host directories other than `$INFORMIXDIR/etc`, set host **INFORMIXSQLHOSTS** environment variables to the appropriate `sqlhosts` file location.
   b. If Connection Managers or clients are installed on hosts where database servers are not installed, create a `sqlhosts` file on each host, and then set each Connection Manager's or client's host **INFORMIXSQLHOSTS** environment variable to the location of the `sqlhosts` file.
3. If Connection Managers, application servers, or database servers are on an untrusted network, complete the following steps:
   a. Create a password file.
   b. Encrypt the password file by running the **onpassword** utility.
   c. Distribute the password file to the database servers that Connection Managers connect to. If the database servers are on other operating systems, distribute the unencrypted password file, and then encrypt it on the other operating systems.
4. On Connection Manager hosts, create a configuration file for each installed Connection Manager.
   a. If the configuration file is in a directory other than `$INFORMIXDIR/etc`, set the **CMCONFIG** environment variable to specify the file's location.

b. If you define a service-level agreement that uses a transaction-latency or apply-failure redirection policy, start quality of data (QOD) monitoring by running the **cdr define qod** and **cdr start qod** commands.

5. Set `onconfig` parameters on the cluster database servers.
   a. If Connection Managers control failover for a high-availability cluster, set the DRAUTO configuration parameter to 3 on all managed cluster database servers, and set the HA_FOC_ORDER configuration parameter on the primary server of each cluster to a failover order.
   b. On each database server that uses multiple ports, set the DBSERVERALIASES configuration parameter to the alias names listed in Connection Manager and database server `sqlhosts` file entries.
6. Optional: Configure the `cmalarmprogram` script that is installed with Connection Managers.
7. Run the **oncmsm** utility to start Connection Managers.

Related concepts:

"The sqlhosts information" on page 2-18

Creating Connection Manager configuration files

To configure a Connection Manager, you must create a configuration file, and then load the configuration file by running the **oncmsm** utility.

A Connection Manager configuration file consists of two parts:

- The header, which contains Connection Manager parameters that are specific to the Connection Manager.
- The body, which consists of one or more connection unit sections that contain parameters and attributes that are specific to the defined connection units.

The following steps apply to all connection-unit types:

1. Create an ASCII text file in the `$INFORMIXDIR/etc` directory of the host the Connection Manager is installed on.
2. On the first line of the file, specify the NAME parameter, followed by a name for the Connection Manager. Connection Manager names must be unique in the domain of the connection units that are managed. For example:

   ```
   NAME my_connection_manager_1
   ```
3. Specify optional header parameters. For example:

   ```
   NAME my_connection_manager_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm_1.log
   ```
4. Create the body of the configuration file by specifying at least one connection unit type followed by the name of the connection unit, and then opening and closing braces. For example:

   ```
   NAME my_connection_manager_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

   CLUSTER my_cluster
   {
   }
   ```
5. Set the connection unit's INFORMIXSERVER parameter to the `sqlhosts` file entries for connection-unit participants. For example:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
    INFORMIXSERVER group_name
}
```

6. If the Connection Manager manages connection requests, specify SLA parameters, SLA names, and DBSERVER attributes. For example:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
   INFORMIXSERVER group_name
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=HDR,SDS,RSS
}
```

7. Specify the FOC parameter and PRIORITY attribute. If the Connection Manager manages failover, specify the ORDER attribute, as well. For example:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
   INFORMIXSERVER group_name
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=HDR,SDS,RSS
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

8. Specify optional SLA parameter attributes. For example:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
   INFORMIXSERVER group_name
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=(HDR,SDS,RSS) \
             POLICY=ROUNDROBIN
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

9. If the Connection Manager manages more than one connection unit, add the other connection units to the body of the configuration file. For example:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
   INFORMIXSERVER group_name
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=(HDR,SDS,RSS) \
             POLICY=ROUNDROBIN
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

```
CLUSTER my_cluster_2
{
   INFORMIXSERVER group_name_2
   SLA sla_3 DBSERVERS=PRI
   SLA sla_4 DBSERVERS=(HDR,SDS) \
             POLICY=ROUNDROBIN
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

Parameters and format of the Connection Manager configuration file

The following example shows the format of the Connection Manager configuration file, and shows which parameters and attributes can be set for each connection-unit type.

```
#****************************** HEADER *******************************

NAME connection_manager_instance_name

# Optional Parameters
MACRO name_1=value
MACRO name_2=value
MACRO name_n=value_n
.
.
.
LOCAL_IP ip_address_list
LOG value
LOGFILE path_and_filename
DEBUG value
CM_TIMEOUT seconds
EVENT_TIMEOUT seconds
SECONDARY_EVENT_TIMEOUT seconds
SQLHOSTS value

#******************************** BODY *******************************

# Replicate set connection-unit example

REPLSET unit_name_1
{
   INFORMIXSERVER sqlhosts_group_names_list

   #Optional Parameters and Attributes
   SLA sla_name_1 DBSERVERS=value_list \

                  #Optional SLA Attributes
                  MODE=value \
                  USEALIASES=value \
                  POLICY=value \
                  WORKERS=number_of_threads \
                  HOST=host_name \
                  NETTYPE=network_protocol \
                  SERVICE=service_name \
                  SQLHOSTSOPT="options"
   SLA sla_name_2 DBSERVERS=value_list ...
   SLA sla_name_n DBSERVERS=value_list ...
   .
   .
   .
}

# High-availability cluster connection-unit example

CLUSTER unit_name_2
```

```
{
   INFORMIXSERVER sqlhosts_group_name
   FOC ORDER=value \
       PRIORITY=value \
       TIMEOUT=value

   #Optional Parameters and Attributes
   SLA sla_name_1 DBSERVERS=value_list \

                  #Optional SLA Attributes
                  MODE=value \
                  USEALIASES=value \
                  POLICY=value \
                  WORKERS=number_of_threads \
                  HOST=host_name \
                  NETTYPE=network_protocol \
                  SERVICE=service_name \
                  SQLHOSTSOPT="options"
   SLA sla_name_2 DBSERVERS=value_list ...
   SLA sla_name_n DBSERVERS=value_list ...
.
.
.
   CMALARMPROGRAM path_and_filename
}

# Server set connection-unit example

SERVERSET unit_name_3
{

   INFORMIXSERVER sqlhosts_group_names_and_standalone_servers_list

   #Optional Parameter and Attributes
   SLA sla_name_1 DBSERVERS=value_list \

                  #Optional SLA Attributes
                  MODE=value \
                  USEALIASES=value \
                  POLICY=value \
                  WORKERS=number_of_threads \
                  HOST=host_name \
                  NETTYPE=network_protocol \
                  SERVICE=service_name \
                  SQLHOSTSOPT="options"
   SLA sla_name_2 DBSERVERS=value_list ...
   SLA sla_name_n DBSERVERS=value_list ...
   .
   .
   .
}

# Grid connection-unit example

GRID unit_name_4
{
   INFORMIXSERVER server_list

   #Optional Parameter and Attributes
   SLA sla_name_1 DBSERVERS=value_list \

                  #Optional SLA Attributes
                  MODE=value \
                  USEALIASES=value \
                  POLICY=value \
                  WORKERS=number_of_threads \
                  HOST=host_name \
```

```
                NETTYPE=network_protocol \
                SERVICE=service_name \
                SQLHOSTSOPT="options"
   SLA sla_name_2 DBSERVERS=value_list ...
   SLA sla_name_n DBSERVERS=value_list ...
   .
   .
   .
}

# Connection Unit n
connection_unit_type unit_name_n
{
   INFORMIXSERVER values
   .
   .
   .
}
#*********************************************************************
```

Tip: For increased readability, break long configuration-file lines by using a backslash (\ ) line-continuation character.

The following example shows a macro definition that uses two text-file lines, but is read as a single line:

```
MACRO servers=node1,node2,node3,node4,node5,node6,node7,node8, \
              node9,node10,node11,node12,node13,node14,node15
```

CMALARMPROGRAM Connection Manager configuration parameter:

The CMALARMPROGRAM parameter specifies the path and file name of a program or script to run if failover processing encounters an error.

Syntax:

Usage

The CMALARMPROGRAM parameter is optional, and is supported by the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

If the Connection Manager cannot find a server capable of receiving failover, it searches for ORDER-attribute servers at increasing intervals, up to 60 seconds, for a maximum of two days. If failover processing fails after eight attempts, the Connection Manager calls the program that is specified by the CMALARMPROGRAM parameter. The first eight failover attempts take approximately one minute.

Before you can use the `cmalarmprogram` script, you must edit the file, and set the script parameters.

The ALARMADMIN and ALARMPAGER parameters determine the level of Connection Manager event alarms that are sent to specified email addresses.

Table 23-1. Connection Manager event-alarm levels

Level of Connection Manager event alarm	Alarm type
0 (default)	None
1	Unimportant informational alarms
2	Informational alarms
3	Alarms requiring attention
4	Emergency alarms
5	Fatal error alarms

Example

In the following example, cmalarmprogram.sh is called if failover processing fails after eight attempts:

```
CMALARMPROGRAM $INFORMIXDIR/etc/cmalarmprogram.sh
```

Related reference:

"FOC Connection Manager configuration parameter" on page 23-11

CM_TIMEOUT Connection Manager configuration parameter:

The CM_TIMEOUT parameter specifies the number of seconds that a cluster of database servers waits to receive events from the failover-arbitrator Connection Manager. If the specified period elapses with no events received by the cluster, the primary server promotes the Connection Manager with the highest priority to the role of failover arbitrator.

Syntax:

Usage

The CM_TIMEOUT parameter is optional, and applies to CLUSTER connection units.

If the CM_TIMEOUT parameter is not specified, the timeout is 60 seconds.

Example

In the following example, if 100 seconds elapse without the servers of a high-availability cluster receiving any events from the current failover arbiter, the primary server of the cluster promotes an active Connection Manager to the role of failover arbiter:

```
CM_TIMEOUT 100
```

CLUSTER Connection Manager configuration parameter:

The CLUSTER parameter specifies that a connection unit is composed of a high-availability cluster, and specifies a name for that connection unit.

Syntax:

```
├─CLUSTER─connection_unit_name──────────────────────────────────────────┤
```

Usage

Each CLUSTER parameter value must be unique within the Connection Manager configuration file.

CLUSTER parameter values cannot use multibyte characters.

CLUSTER connection units can use the following redirection policies:

- Round-robin
- Secondary apply backlog
- Workload

Each CLUSTER connection-unit definition that references a specific high-availability cluster must have a unique PRIORITY attribute value. For example, the following high-availability cluster, composed of **server_1** and **server_2**, must have unique PRIORITY values in each definition.

my_connection_manager_1's configuration file:

```
NAME my_connection_manager_1
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_1.log

CLUSTER my_cluster
{
   INFORMIXSERVER server_1,server_2
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

my_connection_manager_2's configuration file:

```
NAME my_connection_manager_2
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm_2.log

CLUSTER my_cluster
{
   INFORMIXSERVER server_1,server_2
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

Example 1: Specifying a high-availability cluster as a CLUSTER connection unit

In the following example, a high-availability cluster composed of the servers in **my_server_group** is specified as the CLUSTER connection unit **my_cluster**:

```
CLUSTER my_cluster
{
   INFORMIXSERVER my_server_group
```

```
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=ENABLED
       PRIORITY=1
}
```

DEBUG Connection Manager configuration parameter:

The DEBUG parameter specifies whether logging of SQL and ESQL/C error messages is enabled or disabled.

Syntax:

Table 23-2. Values for the DEBUG Connection Manager configuration parameter

DEBUG parameter value	Description
1	Enables logging of SQL and ESQL/C error messages.
0 (default)	Disables logging of SQL and ESQL/C error messages.

Usage

The DEBUG parameter is optional, and applies to the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

Debug messages are created in the location that is specified by the LOGFILE parameter in the Connection Manager configuration file. If the LOGFILE parameter is not specified, then the Connection Manager creates $INFORMIXDIR/tmp/*connection_manager_name.pid*.log.

Connection Manager debug logging is enabled in the configuration file; you cannot enable debug mode from the command line.

Example

In the following example, a my_log is created in $INFORMIXDIR/tmp and logging of SQL and ESQL/C error messages is enabled.

```
LOGFILE $INFORMIXDIR/tmp/my_log
DEBUG 1
```

EVENT_TIMEOUT Connection Manager configuration parameter:

The EVENT_TIMEOUT parameter specifies the number of seconds that must elapse with no primary-server events before the active-arbiter Connection Manager starts failover processing. The Connection Manager waits for primary-server events or notifications from secondary servers that the primary server is offline. A primary-server event is an indication from the primary server that the server is still functioning, such as a sent performance-statistics or administration messages, or node-changes.

Syntax:

Table 23-3. Values for the EVENT_TIMEOUT Connection Manager configuration parameter

EVENT_TIMEOUT parameter value	Description
-1	Connection Manager waits indefinitely
0 to 30	Connection Manager waits 30 seconds.
> 30	Connection Manager waits the specified number of seconds.

Usage

The EVENT_TIMEOUT parameter is optional, and applies to CLUSTER connection units.

If EVENT_TIMEOUT is not specified in the Connection Manager configuration file, the Connection Manager waits 60 seconds for primary-server events or notifications from secondary servers that the primary server is offline before it starts failover processing.

Example

In the following example, the active-arbiter Connection Manager begins failover processing after 200 seconds elapse with no primary-server events.

```
EVENT_TIMEOUT 200
```

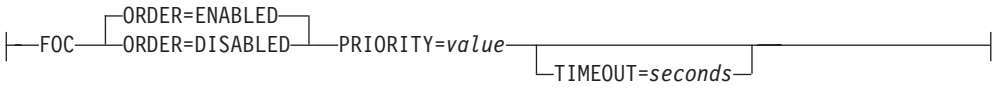
Related reference:

"FOC Connection Manager configuration parameter"

FOC Connection Manager configuration parameter:

The FOC parameter and attributes specify the failover configuration for high-availability clusters, and specify the priority of the connection between the Connection Manager and the primary server.

Syntax:



Attributes of the FOC Connection Manager configuration parameter

The FOC parameter is required by CLUSTER connection units.

Table 23-4. Attributes of the FOC parameter

Attribute of the FOC parameter	Description
ORDER	Disables automatic failover or specifies that the value of the primary server's HA_FOC_ORDER configuration parameter is used for automatic failover. If the ORDER attribute is not specified, the value of the primary server's HA_FOC_ORDER configuration parameter is used for automatic failover. If the ORDER attribute is not specified, and the primary server's HA_FOC_ORDER configuration parameter is not set, the failover order is SDS, HDR, and then RSS.
PRIORITY	Specifies the priority of connections between Connection Managers and the primary server of a cluster. The value must be a positive integer and unique among all Connection Manager CLUSTER units specified for a high-availability cluster. The lower the number, the higher the priority. For CLUSTER connection units, a PRIORITY value must be specified, even if ORDER is set to DISABLED.
TIMEOUT	Specifies the number of additional seconds the Connection Manager waits for primary-server events before the Connection Manager begins failover processing. The TIMEOUT attribute value applies after the EVENT_TIMEOUT parameter value is exceeded. For example, if the EVENT_TIMEOUT parameter is set to 60 and the TIMEOUT value is set to 10, 70 seconds of no primary server events must elapse before failover can begin. If the TIMEOUT attribute is not specified, failover begins immediately after the amount of time that is specified by the EVENT_TIMEOUT parameter value is exceeded.

Values for the ORDER attribute of the FOC Connection Manager configuration parameter

Table 23-5. Values for the ORDER attribute of the FOC Connection Manager configuration parameter.

ORDER attribute value	Description
ENABLED	Specifies that the Connection Manager can perform automatic failover. The Connection Manager uses the value of the primary server's onconfig file HA_FOC_ORDER configuration parameter to determine failover sequence. If the ORDER attribute is set to ENABLED, but the primary server's HA_FOC_ORDER configuration parameter is not specified, the failover order is SDS, HDR, and then RSS.

ORDER Usage

The ORDER attribute specifies if automatic failover is enabled. If failover cannot complete, the Connection Manager reattempts failover at increasing intervals, up to 60 seconds, for a maximum of two days.

To modify the number of seconds that must elapse before the active-arbiter Connection Manager starts failover processing, set the EVENT_TIMEOUT parameter.

If you configure failover, you can set the CMALARMPROGRAM parameter to specify a program or script to run if failover processing cannot complete after eight tries.

Important: If automatic failover is enabled, and failover processing begins, you must not manually restart the failed primary server until failover processing completes.

PRIORITY Usage

If you install Connection Managers on application-server hosts, you can use the PRIORITY attribute to prioritize the connections between the application servers and the primary server of a cluster.

If the network connection between a specific Connection Manager and primary database server fails, PRIORITY determines whether the Connection Manager starts failover. If failover would promote a database server that cannot communicate with an active, higher-priority Connection Manager, then failover does not occur. If failover would promote a database server that cannot communicate with an active, lower-priority Connection Manager, failover can occur.

Example 1: Configuring multiple Connection Managers for failover

In the following example, the HA_FOC_ORDER configuration parameter in the **my_primary_server_1** `onconfig` file is set:

```
HA_FOC_ORDER SDS,HDR,RSS
```

Three Connection Managers are installed and have the following configuration files:

`cm_configuration_file_1`

```
NAME my_connection_manager_1

CLUSTER my_cluster_1
{
   INFORMIXSERVER my_server_group_1
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=1
   CMALARMPROGRAM $INFORMIXDIR/etc/cmalarmprogram.sh
}
```

`cm_configuration_file_2`

```
NAME my_connection_manager_2

CLUSTER my_cluster_1
{
   INFORMIXSERVER my_server_group_1
   SLA sla_2 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=2
   CMALARMPROGRAM $INFORMIXDIR/etc/cmalarmprogram.sh
}
```

```
cm_configuration_file_3
NAME my_connection_manager_3

CLUSTER my_cluster_1
{
   INFORMIXSERVER my_server_group_1
   SLA sla_3 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=3
   CMALARMPROGRAM $INFORMIXDIR/etc/cmalarmprogram.sh
}
```

When the connection managers are initialized, they each search their hosts's sqlhosts file for a **my_server_group_1** entry and connect with the servers that are in the group. The value of the HA_FOC_ORDER configuration parameter in the onconfig file of the primary server in **my_server_group_1** is set for each of the Connection Manager, so that all the Connection Managers have a consistent failover policy. The value of the primary server's HA_FOC_ORDER configuration parameter replaces the values of the HA_FOC_ORDER configuration parameters in the onconfig files of all secondary servers in the cluster. This process maintains failover-order consistency if the primary server fails and then a Connection Manager restarts.

If a Connection Manager detects that the primary server failed, it first attempts to convert the most suitable SD secondary server to the primary server. If no SD secondary server is available, the Connection Manager attempts to convert the HDR secondary server into the primary server. If the HDR secondary server is not available, the Connection Manager attempts to convert the most suitable RS secondary server to the primary server.

If failover processing fails after eight attempts, the Connection Manager calls $INFORMIXDIR/etc/cmalarmprogram.sh.

Example 2: Configuring multiple Connection Managers to prioritize the connections between application servers and the primary server of a cluster

In the following example, the HA_FOC_ORDER configuration parameter in the **my_primary_server_2** onconfig file is set:

```
HA_FOC_ORDER HDR,RSS
```

The two Connection Managers, **my_important_app_cm** and **my_non_essential_app_cm**, are installed and configured:

```
cm_configuration_file_4
NAME important_app_cm
LOCAL_IP 192.0.2.0, 192.0.2.256

CLUSTER my_cluster
{
   INFORMIXSERVER my_server_group_1
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=1
}

cm_configuration_file_5
NAME non_essential_app_cm
LOCAL_IP 192.0.2.257, 192.0.2.258
```

```
CLUSTER my_cluster
{
   INFORMIXSERVER my_server_group_1,
   SLA sla_2 DBSERVERS=ANY
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

- The network state is:
- **important_app_cm** can connect to the primary server and RS secondary server, but cannot connect to the HDR secondary server.
- **non_essential_app_cm** can connect to the primary server and the HDR server, but cannot connect to the RS secondary server.

If the network between **non_essential_app_cm** and **my_primary_server** goes down, **non_essential_app_cm** attempts failover to the HDR secondary server. Because **important_app_cm** cannot connect to the HDR secondary server, and has higher priority than **non_essential_app_cm**, the failover attempt is blocked.

If, instead, the network between **important_app_cm** and **my_primary_server** goes down, **important_app_cm** attempts failover to the HDR secondary server. Because **important_app_cm** cannot connect to the HDR secondary server, it then attempts failover to the RS secondary server. **non_essential_app_cm** cannot connect to the RS secondary server, but it has a lower priority than **important_app_cm**, so failover occurs.

In both situations, the connection between **important_app_cm** and the primary server is prioritized over the connection between **non_essential_app_cm** and the primary server. Because the Connection Managers are on the application-server hosts, the connections between the application servers and the primary server are, essentially, prioritized.

Related tasks:

"Example of configuring connection management for prioritizing connections and network monitoring" on page 23-76

Related reference:

"LOCAL_IP Connection Manager configuration parameter" on page 23-17

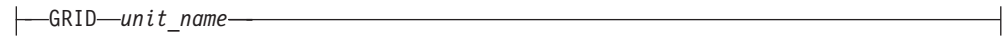
"CMALARMPROGRAM Connection Manager configuration parameter" on page 23-7

"EVENT_TIMEOUT Connection Manager configuration parameter" on page 23-10

GRID Connection Manager configuration parameter:

The GRID parameter specifies that a connection unit is a grid, and specifies the name of the grid.

Syntax:



Usage

Each connection-unit name must be unique within the Connection Manager configuration file.

GRID connection units can use the following redirection policies:

- Apply failure
- Round-robin
- Transaction latency
- Workload

Grid names can use multibyte characters.

Example

In the following example, the grid connection unit named **my_grid** is defined:

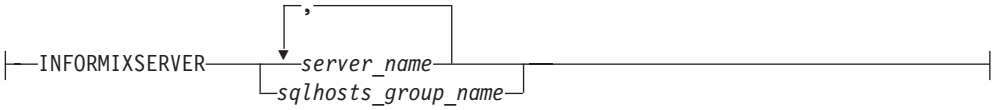
```
GRID my_grid
{
   INFORMIXSERVER my_server_group_1,my_server_group_2
   SLA sla_1 DBSERVERS=ANY
}
```

INFORMIXSERVER Connection Manager configuration parameter:

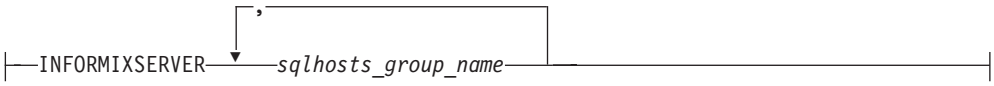
The INFORMIXSERVER parameter specifies database servers or Enterprise Replication nodes that the Connection Manager connects to after it starts. The values of the INFORMIXSERVER parameter are also used for providing database server information for service-level agreements when a Connection Manager's SQLHOSTS parameter is set to `REMOTE` or `LOCAL+REMOTE`.

Syntax

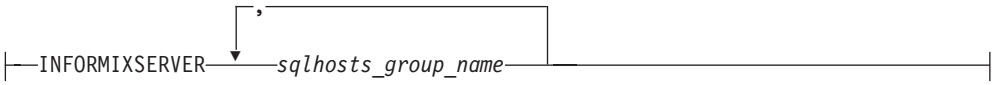
CLUSTER connection unit:



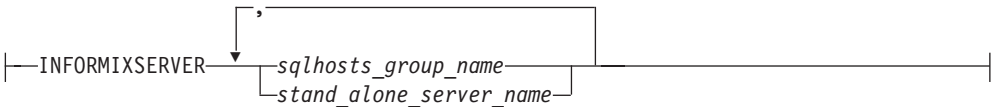
GRID connection unit:



REPLSET connection unit:



SERVERSET connection unit:



Usage

The INFORMIXSERVER parameter is required, and is supported by the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

For Enterprise Replication domains, list group names for the nodes that receive client requests from the Connection Manager.

For server-set replication domains, list group names for the nodes that receive client requests from the Connection Manager. List standalone server names, as well.

For high-availability clusters, list the group name for the cluster, or list the names of database servers.

Example

In the following example, the Connection Manager connects to all database servers that are members of the **my_server_group** group in the Connection Manager `sqlhosts` file.

```
NAME my_connection_manager


CLUSTER my_cluster
{
   INFORMIXSERVER my_server_group
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

Related reference:

"Group information" on page 2-31

LOCAL_IP Connection Manager configuration parameter:

The LOCAL_IP parameter specifies IP addresses to monitor on the computer that is running the Connection Manager. The LOCAL_IP parameter is used with the FOC parameter's PRIORITY attribute to determine if database-failover occurs during a partial network failure.

Syntax:

Usage

The LOCAL_IP parameter is optional, and applies to CLUSTER connection units.

You must list the IP address of each network interface card to be monitored.

Example

In the following example, the Connection Manager monitors the network connection between its host and the primary server of a cluster through the network interface cards that have IP addresses of 192.0.2.0 and 192.0.2.1:

```
LOCAL_IP 192.0.2.0,192.0.2.1
```

Related tasks:

"Example of configuring connection management for prioritizing connections and network monitoring" on page 23-76

Related reference:

"FOC Connection Manager configuration parameter" on page 23-11

LOG Connection Manager configuration parameter:

The LOG parameter specifies logging for Connection Manager modes.

Syntax:

Table 23-6. Values for the LOG Connection Manager configuration parameter

LOG parameter value	Description
0 (default)	Specifies that logging is turned off.
1	The Connection Manager logs proxy-mode and redirect-mode SLA information.
2	The Connection Manager logs proxy-mode SLA information only. Data send-and-receive activities between clients and the Connection Manager is logged.
3	The Connection Manager logs proxy-mode SLA information only. Data content between clients and the Connection Manager is logged.

Usage

The LOG parameter is optional, and applies to the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

Set the LOGFILE parameter to specify where log files are stored. If the LOGFILE parameter is not set, the Connection Manager creates a log file in the `$INFORMIXDIR/tmp` directory, with a name of *`connection_manager_name.process_ID`*`.log`.

Example

In the following example, the Connection Manager logs proxy-mode and redirect-mode SLA information to `$INFORMIXDIR/tmp/my_cm.log`.

```
LOG 1
LOGFILE $INFORMIXDIR/tmp/my_cm.log
```

LOGFILE Connection Manager configuration parameter:

The LOGFILE parameter specifies the name and location of the Connection Manager log file.

Syntax:

Usage

The LOGFILE parameter is optional, and applies to the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

The path and file name of the log file display when the Connection Manager is started, and the log file is continuously updated with status information while the Connection Manager is running. If multiple Connection Managers are installed on the same host, specify a different path and file name for each Connection Manager.

Make sure that the directory for the log file exists, and that access to the file is enabled for the user that starts the Connection Manager.

If the LOGFILE parameter is not set, the Connection Manager creates a log file in the `$INFORMIXDIR/tmp` directory, with a name of *connection_manager_name.process_ID*`.log`.

Example

In the following example, the Connection Manager logs proxy-mode SLA information about data send-and-receive activities between clients and the Connection Manager. The information is logged to `$INFORMIXDIR/tmp/my_cm.log`.

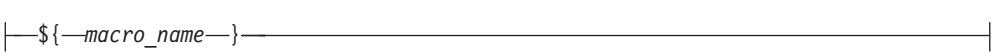
```
LOG 2
LOGFILE $INFORMIXDIR/tmp/my_cm.log
```

MACRO Connection Manager configuration parameter:

The MACRO parameter specifies the name of a macro and a value that can be reused in the Connection Manager configuration file.

Macro definition format:

Macro use format:



Usage

The MACRO parameter is optional, and applies to the Connection Manager configuration file.

A macro can contain spaces, but not line breaks.

The MACRO parameter can be set multiple times to create multiple macros.

After a macro is defined, it can be used in other macros. For example:

```
MACRO WA=wa_server_1,wa_server_2,wa_server_3,wa_server_4
MACRO OR=or_server_1,or_server_2,or_server_3,or_server_4
MACRO ID=id_server_1,id_server_2,id_server_3,id_server_4
MACRO PNW=${WA),${OR),${ID)
```

Example 1: Using a macro in a service-level agreement

In the following example, the macro **CA** contains the names of eight servers.

```
NAME my_connection_manager_1
MACRO CA=ca_server_1,ca_server_2,ca_server_3,ca_server_4, \
         ca_server_5,ca_server_6,ca_server_7,ca_server_8

CLUSTER my_cluster_1
{
   INFORMIXSERVER group_name_1
   SLA sla_1 DBSERVERS=${CA}
   FOC ORDER=ENABLED PRIORITY=1
}
```

The macro expands in the following way:

```
{
   INFORMIXSERVER group_name_1
   SLA sla_1 DBSERVERS=ca_server_1,ca_server_2,ca_server_3,ca_server_4, \
         ca_server_5,ca_server_6,ca_server_7,ca_server_8
   FOC ORDER=ENABLED PRIORITY=1
}
```

Example 2: Using multiple macros in service-level agreements

In the following example, the macros **WA**, **OR**, and **ID** each contain the names of servers. Macro **ID** also contains parentheses to create a redirection-policy group.

```
NAME my_connection_manager_2
MACRO WA=wa_server_1,wa_server_2,wa_server_3
MACRO OR=or_server_1,or_server_2,or_server_3
MACRO ID=(id_server_1,id_server_2,id_server_3)

CLUSTER my_cluster_2
{
   INFORMIXSERVER group_name_2
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=${WA},${OR}
   SLA sla_3 DBSERVERS=${ID} POLICY=ROUNDROBIN
   FOC ORDER=ENABLED PRIORITY=1
}
```

The macros expand in the following ways:

```
{
   INFORMIXSERVER group_name_2
   SLA sla_1 DBSERVERS=PRI
```

```
   SLA sla_2 DBSERVERS=wa_server_1,wa_server_2,wa_server_3,or_server_1,or_server_2,or_server_3
   SLA sla_3 DBSERVERS=(id_server_1,id_server_2,id_server_3) POLICY=ROUNDROBIN
   FOC ORDER=ENABLED PRIORITY=1
}
```

NAME Connection Manager configuration parameter:

The NAME parameter specifies the name of the Connection Manager instance.

Syntax:

Usage

The NAME parameter is required, and applies to the Connection Manager instance.

The name of the Connection Manager instance is necessary for monitoring, shutting down, or reloading the Connection Manager.

Connection Manager names must be unique in the domain of the connection units that are managed.

Example

In the following example, a Connection Manager instance is named **my_connection_manager**.

```
NAME my_connection_manager
```

REPLSET Connection Manager configuration parameter:

The REPLSET parameter specifies that a connection unit is a replicate set, and specifies the name of the replicate set.

Syntax:

Usage

Each connection-unit name must be unique within the Connection Manager configuration file.

REPLSET connection units can use the following redirection policies:

- Apply failure
- Round-robin
- Transaction latency
- Workload

Replicate set names can use multibyte characters.

Example

In the following example, the replicate-set connection unit named **my_replicate_set** is defined:

```
REPLSET my_replicate_set
{
   INFORMIXSERVER my_group_1,my_group_2
   SLA sla_1 DBSERVERS=ANY
}
```

SECONDARY_EVENT_TIMEOUT Connection Manager configuration parameter:

The SECONDARY_EVENT_TIMEOUT parameter specifies the number of seconds that must elapse with no secondary-server events before the Connection Manager disconnects from a secondary server. A secondary-server event is an indication from a secondary server that the server is still functioning, such as a sent performance-statistics or administration messages.

Syntax:

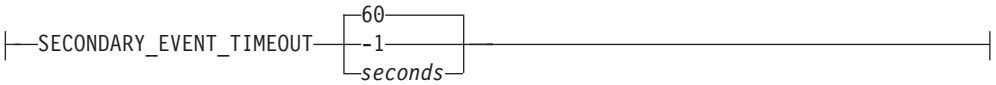


Table 23-7. Values for the SECONDARY_EVENT_TIMEOUT Connection Manager configuration parameter

SECONDARY_EVENT_TIMEOUT parameter value	Description
-1	The Connection Manager waits indefinitely
0 to 30	The Connection Manager waits 30 seconds.
> 30	The Connection Manager waits the specified number of seconds.

Usage

The SECONDARY_EVENT_TIMEOUT parameter is optional, and applies CLUSTER connection units.

If the SECONDARY_EVENT_TIMEOUT parameter is not specified in the Connection Manager configuration file, the Connection Manager waits 60 seconds for secondary-server events before it disconnects from the server.

Example

In the following example, the Connection Manager disconnects from a secondary server after 300 seconds elapse with no secondary-server events.

```
SECONDARY_EVENT_TIMEOUT 300
```

SERVERSET Connection Manager configuration parameter:

The SERVERSET parameter specifies that a connection unit is a server set, and specifies the name of the server set.

Syntax:

Usage

Each connection-unit name must be unique within the Connection Manager configuration file.

SERVERLSET connection units can use the following redirection policies

- Round-robin
- Workload

Server set names cannot use multibyte characters.

Example

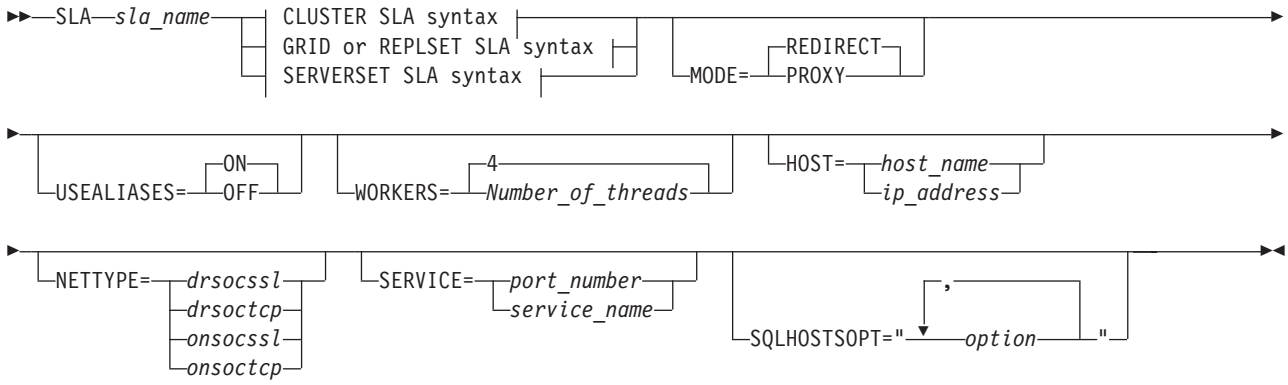
In the following example, the server-set connection unit named **my_server_set** is defined:

```
SERVERSET my_server_set
{
   INFORMIXSERVER my_group_1,my_group_2
   SLA sla_1 DBSERVERS=ANY
}
```

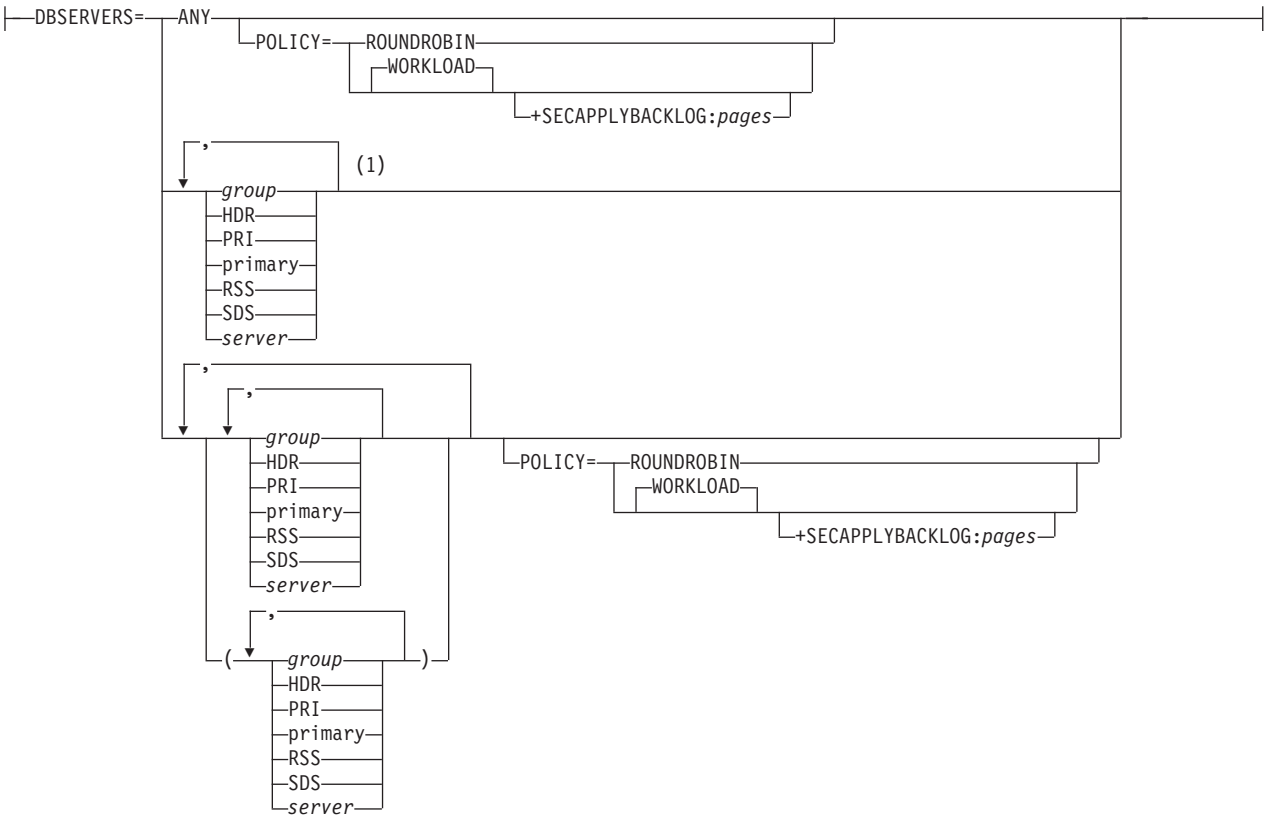
SLA Connection Manager configuration parameter:

The SLA parameter defines service-level agreements that direct client requests to database servers.

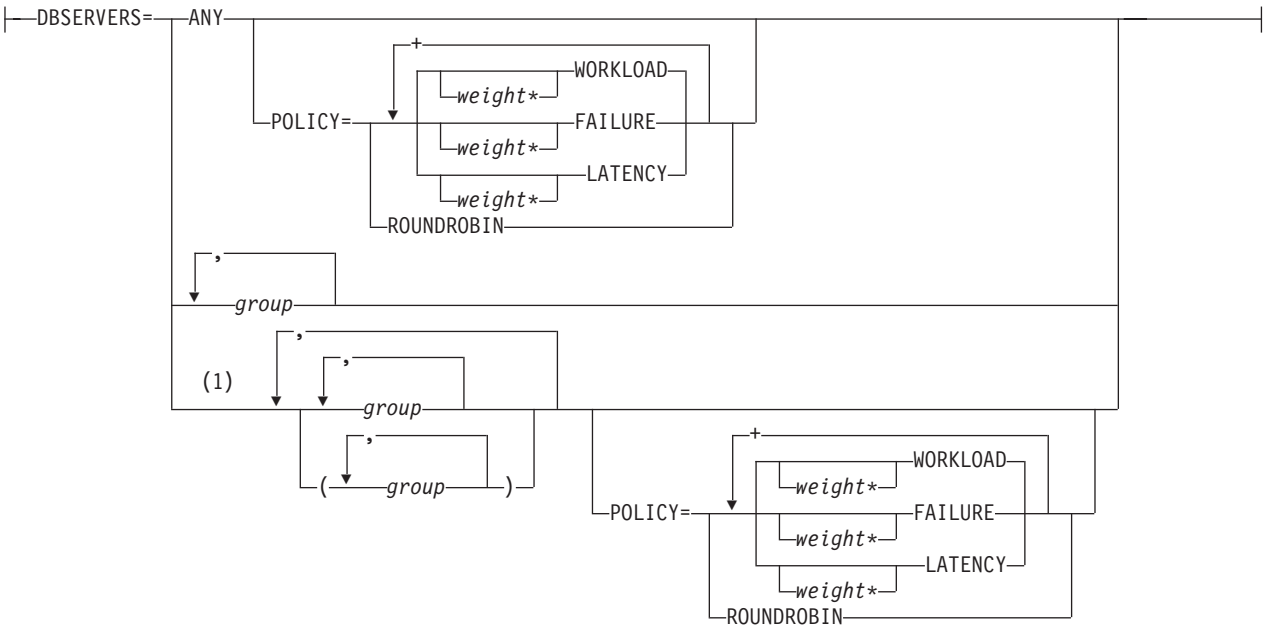
Syntax



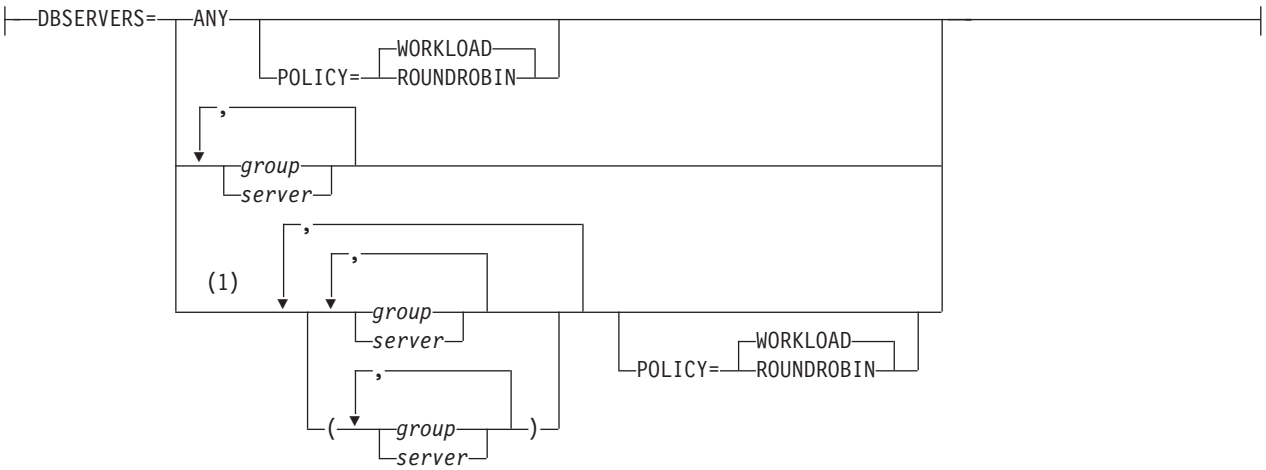
CLUSTER SLA syntax fragment:



GRID or REPLSET SLA syntax fragment:



SERVERSET SLA syntax fragment:



Notes:

1 You must use at least one cluster keyword or a group of values that are enclosed by parentheses if you specify a redirection policy.

SLA parameter attributes

Table 23-8. The attributes of the SLA Connection Manager configuration parameter

Attribute name	Description
DBSERVERS	Specifies servers, server aliases, server groups, or server types for directing connection requests. Use the ANY keyword, the SDS or RSS cluster keywords, or enclose a group of values in parentheses to enable a redirection policy for that group.
HOST	Specifies a database server's host. The value in the SLA is used, rather than the value in the Connection Manager's host `sqlhosts` file.
MODE	Specifies whether connection requests go through the Connection Manager or if the Connection Manager provides connection information to the source of a connection request. The default value is REDIRECT.
NETTYPE	Specifies the network protocol of a database server. The value in the SLA is used, rather than the value in the Connection Manager's host `sqlhosts` file.
POLICY	Specifies how the Connection Manager redirects client connection requests to the servers specified in the DBSERVER attribute. Redirection policy applies to the ANY keyword, the SDS and RSS cluster keywords, and to a group of values that are enclosed in parentheses. Parentheses within parentheses are ignored. The default value is WORKLOAD. Workload, apply-failure, and transaction-latency policies can be given relative weights.
SERVICE	Specifies a database server's port number or service name. The value in the SLA is used, rather than the value in the Connection Manager's host `sqlhosts` file.
SQLHOSTSOPT	Specifies connectivity options for a database server that is specified in a SLA. Enclose all connectivity options in a single pair of quotation marks. The value in the SLA is used, rather than the value in the Connection Manager's host `sqlhosts` file.

Table 23-8. The attributes of the SLA Connection Manager configuration parameter (continued)

Attribute name	Description
USEALIASES	Specifies whether the Connection Manager can redirect client connection requests to database server aliases specified by the DBSERVERALIASES configuration parameter. The default value is ON.
WORKERS	Specifies the number of worker threads that are allocated to the SLA. When a service-level agreement is specified, the Connection Manager creates an SLA listener process to intercept client connection requests. The SLA listener process can have one or more worker threads. The default value is 4.

DBSERVERS attribute values

Table 23-9. Values of the DBSERVERS attribute.

Attribute value	Value
ANY	Specifies that connection requests can be sent to any available database server in the specified cluster, grid, or replicate set. For a SERVERSET connection unit, ANY specifies that connection requests can be sent to any available database server specified by the SERVERSET connection-unit's INFORMIXSERVER parameter. You do not need to enclose ANY in parentheses to apply a redirection policy to it. If no redirection policy is specified, ANY uses the workload redirection policy.
group	Specifies a group entry in the Connection Manager's host sqlhosts file. Connection requests can be sent to the members of the group.
HDR	Is a cluster keyword that specifies that connection requests can be sent to the high-availability data replication server. HDR is supported only by CLUSTER connection units.
PRI or PRIMARY	Is a cluster keyword that specifies that connection requests can be sent to the primary database server. PRI and PRIMARY are supported only by CLUSTER connection units.
RSS	Is a cluster keyword that specifies that connection requests can be sent to remote standalone secondary servers. RSS is supported only by CLUSTER connection units. You do not need to enclose RSS in parentheses to apply a redirection policy to the servers it specifies. If no redirection policy is specified, RSS uses the workload redirection policy.
SDS	Is a cluster keyword that specifies that connection requests can be sent to shared-disk secondary servers. SDS is supported only by CLUSTER connection units. You do not need to enclose SDS in parentheses to apply a redirection policy to the servers it specifies. If no redirection policy is specified, SDS uses the workload redirection policy.
server	Specifies server or alias entry in the Connection Manager's host sqlhosts file. Connection requests can be sent to the server.

MODE attribute values

Table 23-10. Values of the MODE attribute.

Attribute value	Value
PROXY	Specifies that the Connection Manager acts as a proxy server for client connections. Use proxy mode for the following cases: • A firewall is preventing a client application from connecting to database servers. • You do not want to recompile applications are compiled with Data Server Driver for JDBC and SQLJ version 3.5.1 or before, or with IBM Informix Client Software Development Kit (Client SDK) 3.00 or before. Because a proxy-server Connection Manager handles all client/server communication, configure multiple Connection Manager instances, to avoid a Connection Manager becoming a single point of failure. **Note:** For proxy mode, you must set your operating system to allow the maximum number of file descriptors. For example, use the **ulimit** command on UNIX operating systems.
REDIRECT (default)	Specifies that client connections use redirect mode, which configures the Connection Manager to return the appropriate database server name, IP address, and port number to the requesting client application. The client application then uses the returned IP address and port number to connect to the specified database server. Redirection-policy SLAs do not support connections from application that are compiled with Data Server Driver for JDBC and SQLJ version 3.5.1 or before, or with IBM Informix Client Software Development Kit (Client SDK) 3.00 or before.

POLICY attribute values

Table 23-11. Values of the POLICY attribute.

Attribute value	Value
FAILURE	Specifies that connection requests are directed or proxied to the replication server with the fewest apply failures. The apply-failure policy is supported by the following connection units: • REPLSET • GRID To use the apply-failure policy, you must enable quality of data (QOD) monitoring by running the **cdr define qod** and **cdr start qod** commands. To use the apply-failure policy for a grid, the grid must have a replication-enabled table.

Table 23-11. Values of the POLICY attribute. (continued)

Attribute value	Value
LATENCY	Specifies that connection requests are directed or proxied to the replication server with the lowest transaction latency. The transaction-latency policy is supported by the following connection units: • REPLSET • GRID The attribute value does not indicate a specific transaction latency period; the Connection Manager uses a formula with relative values to decide where to redirect client connection requests. To use the transaction-latency policy, you must enable quality of data monitoring by running the **cdr define qod** and **cdr start qod** commands. To use the transaction-latency policy for a grid, the grid must have a replication-enabled table.

ROUNDROBIN	Specifies that connection requests are directed or proxied in a repeating, ordered fashion (round-robin) to a group of servers.
	
	If you use a round-robin policy, the DBSERVERS attribute values are used to create round-robin groups. Servers that are specified more than one time in a DBSERVERS-attribute group value are treated as single participants in a round-robin group. For example, if a SLA has the following definition:
	`SLA sla_1 DBSERVERS=(server_1,server_3,server_1,server_2) \`
	`POLICY=ROUNDROBIN`
	
	The round-robin group participants are:
	• **server_1**
	• **server_2**
	• **server_3**
	
	The round-robin policy is supported by the following connection units:
	• CLUSTER
	• REPLSET
	• GRID
	• SERVERSET
	
	The round-robin policy is supported by the following software:
	• All IBM Informix server versions.
	• Connection Managers from IBM Informix Client Software Development Kit (Client SDK) 3.70.xC8 and later, and 4.10xC2 and later.

Table 23-11. Values of the POLICY attribute. (continued)

Attribute value	Value

SECAPPLYBACKLOG:*number_of_pages*	Specifies that if a secondary server's apply backlog exceeds *number_of_pages*, the Connection Manager does not redirect or proxy new connections to server. For example:
	`SLA sla_1 DBSERVERS=(server_1,server_2,server_3) \`
	`POLICY=SECAPPLYBACKLOG:500`
	The Connection Manager sends connection requests to whichever of **server_1**, **server_2**, or **server_3** has an apply backlog below 500 pages and the lowest workload.
	To view the apply backlogs of all servers in a cluster, run the **onstat -g** cluster command.
	To view the apply backlog for a specific secondary server, run one of the following commands:
	• **onstat -g dri**
	• **onstat -g sds**
	• **onstat -g rss**
	The apply-backlog policy is supported by CLUSTER connection units.
	The apply-backlog policy is supported by the following software:
	• IBM Informix server versions 11.70.xC8 and later, and 12.10.xC2 and later.
	• Connection Managers from IBM Informix Client Software Development Kit (Client SDK) 3.70.xC8 and later, and 4.10xC2 and later.

WORKLOAD (default)	Specifies that connection requests are directed or proxied to the database server with the lowest workload. Workload calculations are based on the number of virtual processors a server has and the number of threads in the server's ready queue. The WORKLOAD policy is supported by the following connection units: • CLUSTER • GRID • REPLSET • SERVERSET

USEALIASES attribute values

Table 23-12. Values of the USEALIASES attribute.

Attribute value	Value
ON (default)	Specifies that the Connection Manager can direct client connection requests to server aliases specified by a database server's DBSERVERALIASES configuration parameter.
OFF	Specifies that the Connection Manager cannot direct client connection requests to server aliases specified by a database server's DBSERVERALIASES configuration parameter.

HOST attribute values

Table 23-13. Values of the HOST attribute.

Attribute value	Value
host_name	Specifies a host name or host alias for a database server.
ip_address	Specifies a TCP/IP address for a database server.

NETTYPE attribute values

Table 23-14. Values of the NETTYPE attribute.

Attribute value	Value
`drsocssl`	Specifies secured sockets layer (SSL) protocol for Distributed Relational Database Architecture (DRDA).
`drsoctcp`	Specifies TCP/IP protocol for Distributed Relational Database Architecture
`onsocssl`	Specifies secured sockets layer protocol.
`onsoctcp`	Specifies sockets with TCP/IP protocol.

SERVICE attribute values

Table 23-15. Values of the SERVICE attribute.

Attribute value	Value
port_number	Specifies a port number.
service_name	Specifies a service name.

Usage

The SLA parameter is optional, and is supported by the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

Client applications use the SLA name to connect to the database servers or database-server types that are specified by the value of the DBSERVERS attribute. For each SLA, a listener thread is installed at the specified port on the server to detect incoming client requests. The SLA parameter can be specified multiple times in the same configuration file; however, each SLA name must be unique.

Example 1: Connection request redirection from a service-level agreement

The following example shows a simple Connection Manager configuration. The configuration specifies a single SLA.

```
NAME my_connection_manager_1

CLUSTER my_cluster_1
{
   INFORMIXSERVER my_server_group_1
   SLA sla_1 DBSERVERS=SDS,HDR,PRI
   FOC ORDER=ENABLED PRIORITY=1
}
```

CONNECT TO @sla_1 connection requests as are directed in the following way:

1. Connect to any available SD secondary servers.
2. If SD secondary servers are unavailable, connect to the HDR secondary server.
3. If the HDR secondary server is unavailable, connect to the primary server.

Example 2: Defining multiple service-level agreements

The following example shows a simple Connection Manager configuration. The configuration specifies two SLAs.

```
NAME my_connection_manager_2

CLUSTER my_cluster_2
{
   INFORMIXSERVER my_server_group_2
   SLA sla_1 DBSERVERS=server_1
   SLA sla_2 DBSERVERS=server_2,server_3
   FOC ORDER=ENABLED PRIORITY=1
}
```

This example configures the Connection Manager for a high-availability cluster and defines two SLAs:

- CONNECT TO @sla_1 connection requests are directed to **server_1**.
- CONNECT TO @sla_2 connection requests are directed to **server_2**. If **server_2** is not available, connection requests are directed to **server_3**.

Example 3: Redirection policies in service-level agreements

The following example shows a Connection Manager configuration that uses a workload-balancing redirection policy. The configuration specifies a single SLA.

```
NAME my_connection_manager_3

CLUSTER my_cluster_3
{
   INFORMIXSERVER my_server_group_3
   SLA sla_1 DBSERVERS=(server_1,server_2) \
             POLICY=WORKLOAD
   SLA sla_2 DBSERVERS=(server_3,server_4,server_5) \
             POLICY=ROUNDROBIN
   SLA sla_3 DBSERVERS=(server_6,server_7,server_8) \
             POLICY=ROUNDROBIN+SECAPPLYBACKLOG:400
   FOC ORDER=ENABLED PRIORITY=1
}
```

- CONNECT TO @sla_1 connection requests are directed to whichever of **server_1** and **server_2** has the lowest workload. WORKLOAD is the default redirection policy, so specifying POLICY=WORKLOAD in the SLA is not required.
- CONNECT TO @sla_2 connection requests are directed round-robin to **server_3**, **server_4** and **server_5**.
- CONNECT TO @sla_3 connection requests are directed round-robin to **server_6**, **server_7** and **server_8**. If a server's apply backlog is 400 pages or greater, that server is ignored in the round-robin order and does not receive connection requests until its apply backlog falls below 400 pages.

Example 4: Proxy and redirect mode in service-level agreements

In redirect mode, the Connection Manager responds to client redirection requests by returning a specified database server's IP address and port number to the client application. The client application then uses the IP address and port number to

connect to the database server. In proxy mode, the Connection Manager acts as a proxy server, and client requests are routed through the Connection Manager. Redirect mode is the default if no SLA mode is specified.

```
NAME my_connection_manager_4

CLUSTER my_cluster_4
{
   INFORMIXSERVER my_server_group_4
   SLA sla_1 DBSERVERS=ANY \
             MODE=REDIRECT #Default value, so is not required for the SLA definition
   SLA sla_2 DBSERVERS=ANY \
             MODE=PROXY
   FOC ORDER=ENABLED PRIORITY=1
}
```

- `CONNECT TO @sla_1` connection requests result in the Connection Manager returning the IP address and port number for a cluster server to the client application.
- `CONNECT TO @sla_2` connection requests are directed through the Connection Manager to a cluster server.

Example 5: Adjusting timeout values for the Connection Manager and cluster servers

The following example shows a Connection Manager configuration where default timeout values are changed:

```
NAME my_connection_manager_5
CM_TIMEOUT 300
EVENT_TIMEOUT 45
SECONDARY_EVENT_TIMEOUT 50

CLUSTER my_cluster_5
{
   INFORMIXSERVER my_server_group_5
   SLA sla_1 DBSERVERS=ANY
   FOC ORDER=ENABLED PRIORITY=1
}
```

- If a cluster does not receive any events from the Connection Manager within 300 seconds, the primary server of the cluster promotes the next available Connection Manager to the role of failover arbitrator.
- If the Connection Manager does not receive any events from the primary server within 45 seconds, the Connection Manager begins failover processing, and attempts to promote a secondary server to the primary server.
- If the Connection Manager does not receive any events from a secondary server within 50 seconds, the Connection Manager disconnects from the secondary server.

Example 6: Macros and workload balancing in service-level agreements

The following example shows a Connection Manager configuration that uses defined macros in SLAs.

```
NAME my_connection_manager_6
MACRO CA=ca_server_1,ca_server_2,ca_server_3
MACRO NY=ny_server_1,ny_server_2,ny_server_3

REPLSET my_replicate_set_1
{
   INFORMIXSERVER my_er_group_1,my_er_group_2
```

```
   SLA sla_1 DBSERVERS=${CA}
   SLA sla_2 DBSERVERS=(${NY}) \
             POLICY=ROUNDROBIN
}
```

In this example, two macros are defined:

- CA, which is composed of **ca_server_1**, **ca_server_2**, and **ca_server_3**.
- NY, which is composed of **ny_server_1**, **ny_server_2**, and **ny_server_3**.

The Connection Manager redirects client connection requests as follows:

- CONNECT TO @sla_1 connection requests are directed to **ca_server_1**. If **ca_server_1** is unavailable, connection requests are directed to **ca_server_2**. If **ca_server_2** is also unavailable, connection requests are directed to **ca_server_3**.
- CONNECT TO @sla_2 connection requests are directed round-robin to **ny_server_1**, **ny_server_2**, and **ny_server_3**.

Example 7: Quality of data redirection policies in service-level agreements

The following example shows a Connection Manager configuration that uses transaction-latency and apply-failure redirection policies to direct connection requests in a grid. Quality-of-data (QOD) monitoring is turned on with the **cdr define qod** and **cdr start qod** commands.

```
NAME my_connection_manager_7

GRID my_grid_1
{
   INFORMIXSERVER my_server_group_1,my_server_group_2,my_server_group_3
   SLA sla_1 DBSERVERS=ANY \
             POLICY=LATENCY
   SLA sla_2 DBSERVERS=ANY \
             POLICY=FAILURE
   SLA sla_3 DBSERVERS=ANY \
             POLICY=2*Failure+LATENCY
}
```

- CONNECT TO @sla_1 connection requests are directed to the server with the lowest transaction latency.
- CONNECT TO @sla_2 connection requests are directed to the server with the lowest number of apply failures.
- CONNECT TO @sla_3 connection requests are directed to the server with the lowest number of apply failures and lowest transaction latency. The smallest apply-failure count is twice as important as low transaction latency in the Connection Manager's calculations.

Example 8: sqlhosts connectivity information in service-level agreements

The following example shows a Connection Manager configuration that uses attribute values instead of the values in its host sqlhosts file for directing connection requests.

```
NAME my_connection_manager_8

SERVERSET my_server_set
{
   INFORMIXSERVER server_1,server_2,server_3
   SLA sla_1 DBSERVERS=server_1 \
             NETTYPE=onsoctcp
             HOST=host_1 \
             SERVICE=port_1 \
   SLA sla_2 DBSERVERS=server_2 \
```

```
            NETTYPE=onsoctcp
            HOST=host_2 \
            SERVICE=port_2 \
}
```

The Connection Manager uses the values of the HOST, SERVICE, and NETTYPE attributes, rather than the values in its host `sqlhosts` file for directing connection requests.

Example 9: Controlling connection-requests with aliases

The following example shows a Connection Manager configuration that specifies which database-server aliases SLAs can use.

The `onconfig` file for **server_1** has the following parameter setting:

```
DBSERVERALIASES server_1_alias_1,server_1_alias_2
```

The `sqlhosts` file that **server_1** uses has the following entries:

```
#dbservername      nettype   hostname   servicename  options
my_group_9         group     -          -            e=server_1_alias_2
server_1           onsoctcp  my_host_1  my_port_1    g=my_group_9
server_1_alias_1   onsoctcp  my_host_1  my_port_2    g=my_group_9
server_1_alias_2   onsoctcp  my_host_1  my_port_3    g=my_group_9
```

The Connection Manager configuration file for **server_1** has the following entries:

```
NAME my_connection_manager_9

SERVERSET my_server_set_2
{
   INFORMIXSERVER my_group_9
   SLA sla_1 DBSERVERS=server_1
   SLA sla_2 DBSERVERS=server_1 \
             USEALIASES=OFF
   SLA sla_3 DBSERVERS=server_1_alias_1
   SLA sla_4 DBSERVERS=server_1_alias_1 \
             USEALIASES=OFF
}
```

The Connection Manager directs client requests in the following ways:

- `CONNECT TO @sla_1` and `CONNECT TO @sla_3` requests can be directed to **server_1**, through **my_port_1**, **my_port_2**, or **my_port_3**.
- `CONNECT TO @sla_2` requests are directed to **server_1** through **my_port_1** only.
- `CONNECT TO @sla_4` requests are directed to **server_1** through **my_port_2** only.

Related reference:

"Group information" on page 2-31

SQLHOSTS Connection Manager configuration parameter:

The SQLHOSTS parameter specifies where a Connection Manager can search for database servers that are specified by the INFORMIXSERVER parameter and DBSERVERS attribute.

Syntax:

Table 23-16. Values for the SQLHOSTS Connection Manager configuration parameter

SQLHOSTS parameter value	Description
LOCAL	The Connection Manager searches the local sqlhosts file for requested database server instances.
REMOTE	The Connection Manager searches for requested database server instances in remote sqlhosts files. The Connection Manager searches the sqlhosts files of database servers that are specified by a connection-unit's INFORMIXSERVER parameter.
LOCAL+REMOTE (default)	The Connection Manager searches the local sqlhosts file for requested database server instances. If a database server cannot be found, the Connection Manager searches the sqlhosts files of database servers that are specified by a connection-unit's INFORMIXSERVER parameter.

Usage

The SQLHOSTS parameter is optional, and applies to the following connection units:

- CLUSTER
- GRID
- REPLSET
- SERVERSET

The SQLHOSTS option is useful when you want to restrict client applications from accessing one or more database servers in a high-availability cluster.

Example

In the following example, the SQLHOSTS parameter is used to limit the servers that the Connection Manager connects to.

```
NAME my_connection_manager
SQLHOSTS LOCAL

CLUSTER my_cluster
{
   INFORMIXSERVERS my_servers
   SLA my_sla DBSERVERS=server_1,server_2,server_3,server_4
   FOC ORDER=ENABLED PRIORITY=1
}
```

The local sqlhosts file has the following entries:

```
#dbservername       nettype    hostname   servicename   options
 my_servers         group      -          -             c=1,e=server_3
 server_1           onsoctcp   host_1     port_1        g=my_servers
 server_2           onsoctcp   host_2     port_2        g=my_servers
 server_3           onsoctcp   host_3     port_3        g=my_servers
```

Remote database-server sqlhosts files have the following entries:

```
#dbservername       nettype    hostname   servicename   options
 my_servers         group      -          -             c=1,e=server_4
 server_1           onsoctcp   host_1     port_1        g=my_servers
 server_2           onsoctcp   host_2     port_2        g=my_servers
 server_3           onsoctcp   host_3     port_3        g=my_servers
 server_4           onsoctcp   host_3     port_3        g=my_servers
```

server_4 is not defined in the local sqlhosts file, and the Connection Manager is configured to not search remote sqlhosts files, so the Connection Manager does not send connection requests to **server_4**.

Related reference:

"Group information" on page 2-31

Modifying Connection Manager configuration files

Use the **oncmsm** utility to load a modified configuration file into a Connection Manager and change the Connection Manager's configuration.

If you are using multiple Connection Managers, you can run the **onstat -g cmsm** command to display the names of Connection Manager instances.

1. Modify the existing Connection Manager configuration file. The default location of the configuration file is the $INFORMIXDIR/etc directory.
2. On the Connection Manager's host, run the **oncmsm** utility with the **r** parameter and the name of the Connection Manager instance. For example:

   ```
   oncmsm -r connection_manager_name
   ```

   Because multiple instances of the Connection Manager can be active at the same time, you must specify the name of the Connection Manager instance.

After you reload a Connection Manager's configuration file, the new configuration immediately takes effect.

Converting older formats of the Connection Manager configuration file to the current format

The Connection Manager configuration file in versions of IBM Informix Client Software Development Kit (Client SDK) before version 3.70.xC3 are incompatible with the current version of the Connection Manager. You must convert configuration files from versions before 3.70.xC3 by running the **oncmsm** utility.

If you have multiple configuration files to convert, combine related SLA definitions into a single configuration file.

You can convert Connection Manager configuration files, even if a Connection Manager is running.

To convert a Connection Manager file to the current format:

1. Log on to the computer on which the updated Client SDK is installed.
2. Run the **oncmsm** utility, specifying old and new configuration file names.

After the configuration file is converted, it can be loaded into a Connection Manager.

Example: Converting a configuration file and loading it into a Connection Manager

For the following example:

- You are using a UNIX operating system.
- You updated a Connection Manager from 3.50.xC9 to 4.10xC1, and must update your configuration file.
- The Connection Manager was using a configuration file that is named configuration_file_old and located in $INFORMIXDIR/etc.

- You want to convert the file to the new format and rename the file to `configuration_file_new`.

Run the following commands:

1. `oncmsm -c configuration_file_old -n configuration_file_new`
2. `oncmsm -c configuration_file_new`

Configuring environments and setting configuration parameters for connection management

Before you start a Connection Manager, you must configure its environment.

To set environment variables:

- For UNIX C use the appropriate shell command. The following examples use C shell (csh).
- For Windows, use the **Environment** tab of the **setnet32** utility.

1. If clients or Connection Managers are installed on hosts where database servers are not installed, set each host's **INFORMIXDIR** environment variable to the directory the client or Connection Manager is installed in. Run the following command:

   `setenv INFORMIXDIR` *path*
2. If clients or Connection Managers are installed on hosts where database servers are not installed, set each host's **INFORMIXSQLHOSTS** environment variable to the location of the `sqlhosts` file that the host uses. Run the following command:

   `setenv INFORMIXSQLHOSTS` *path_and_filename*
3. If a Connection Manager's configuration file is in a directory other than `$INFORMIXDIR/etc`, set the **CMCONFIG** environment variable to the directory. Run the following command:

   `setenv CMCONFIG` *path_and_filename*
4. If Connection Managers control failover for a high-availability cluster, set the `onconfig` file DRAUTO configuration parameter to 3 on all managed cluster database servers. For example:

   ```
   DRAUTO 3
   ```
5. If Connection Managers control failover for a high-availability cluster, set the `onconfig` file HA_FOC_ORDER configuration parameter on the primary server of each cluster to a failover order. For example:

   ```
   HA_FOC_ORDER SDS,HDR,RSS
   ```
6. On each database server that uses multiple ports, set the `onconfig` file DBSERVERALIASES configuration parameter to the alias names listed in Connection Manager and database server `sqlhosts` file entries. For example:

   A Connection Manager's host has the following `sqlhost` file entries:

   ```
   #dbservername   nettype    hostname   servicename   options
    my_servers     group      -          -             c=1,e=server_3
    server_1       onsoctcp   host_1     port_1        s=6,g=my_servers
    server_2       onsoctcp   host_2     port_2        s=6,g=my_servers
    server_3       onsoctcp   host_3     port_3        s=6,g=my_servers

    my_aliases     group      -          -             c=1,e=a_server_3
    a_server_1     onsoctcp   host_1     port_4        g=my_aliases
    a_server_2     onsoctcp   host_2     port_5        g=my_aliases
    a_server_3     onsoctcp   host_3     port_6        g=my_aliases
   ```

   In the `onconfig` file for **server_1**, set the following value:

   ```
   DBSERVERALIASES a_server_1
   ```

   In the `onconfig` file for **server_2**, set the following value:

DBSERVERALIASES a_server_2

In the onconfig file for **server_3**, set the following value:

DBSERVERALIASES a_server_3

Defining sqlhosts information for connection management

You must define sqlhosts network-connectivity information for client applications that connect to Connection Managers, Connection Managers that connect to database servers, and database servers that are part of a Connection Manager connection unit.

If Connection Managers or clients are installed on hosts where database servers are not installed, you must create a sqlhosts file on each host.

Entries in an sqlhosts file can specify connection information for the following connection-unit components:

- Database servers
- Aliases for database servers that are using secure ports
- Connection Manager service-level agreements (SLAs)
- Groups that can contain database servers, database-server aliases, or SLAs.

All database servers that a Connection Manager connects to must be listed in the sqlhosts file that the Connection Manager uses. If the Connection Manager is monitoring a high-availability cluster, the sqlhosts file the Connection Manager uses must contain entries for all cluster servers.

1. Create entries in each database server's host sqlhosts file. You can modify one sqlhosts file, and then distribute it to the hosts of other database servers.
2. Create entries in each Connection Manager's host sqlhosts file. You can create one sqlhosts file, and then distribute it to the hosts of other Connection Managers.
3. Create entries in each client application's host sqlhosts file. You can create one sqlhosts file, and then distribute it to the hosts of other client applications.
4. If a host has multiple database servers that are installed on it, if the sqlhosts file is in a directory other than $INFORMIXDIR/etc, or if you are using a network-connectivity file other than $INFORMIXDIR/etc/sqlhosts, set the host's **INFORMIXSQLHOSTS** environment variable to the location of the sqlhosts file.

If sqlhosts file entries use the s=6 option to define secure ports, use the information in the sqlhosts file to create a password file.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of high-availability clusters

You must define sqlhosts network-connectivity information for connection management of high-availability clusters.

To use a file other than $INFORMIXDIR/etc/sqlhosts on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and database server, create sqlhosts file entries for each database server in the cluster. For example:

```
#dbservername   nettype    hostname   servicename   options
 server_1       onsoctcp   host_1     port_1
 server_2       onsoctcp   host_2     port_2
 server_3       onsoctcp   host_3     port_3
```

2. On the host of each Connection Manager, add a group entry that contains each database server in the cluster, and add group options to the database-server entries. Use the c=1 group-entry option, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

```
#dbservername   nettype    hostname   servicename   options
 my_servers     -          -                        c=1,e=server_3
 server_1       onsoctcp   host_1     port_1        g=my_servers
 server_2       onsoctcp   host_2     port_2        g=my_servers
 server_3       onsoctcp   host_3     port_3        g=my_servers
```

3. On the host of each client application, create an sqlhosts file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

```
NAME connection_manager_1

CLUSTER my_cluster
{
   INFORMIXSERVER my_servers
   SLA sla_primary_1     DBSERVERS=PRI
   SLA sla_secondaries_1 DBSERVERS=SDS,HDR
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

   The second Connection Manager's configuration file has the following entries:

```
NAME connection_manager_2

CLUSTER my_cluster
{
   INFORMIXSERVER my_servers
   SLA sla_primary_2     DBSERVERS=PRI
   SLA sla_secondaries_2 DBSERVERS=SDS,HDR
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

   Add the following entries to each client application's host sqlhosts file:

```
#dbservername       nettype    hostname   servicename   options
 sla_primary_1      onsoctcp   cm_host_1  cm_port_1
 sla_primary_2      onsoctcp   cm_host_2  cm_port_2

 sla_secondaries_2  onsoctcp   cm_host_1  cm_port_3
 sla_secondaries_2  onsoctcp   cm_host_2  cm_port_4
```

4. On the host of each client application, create sqlhosts file entries for each group of SLA entries, and add group options to the SLA entries. Use the c=1 group-entry option, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members.

```
#dbservername       nettype    hostname   servicename  options
 g_primary          group      -          -            c=1,e=sla_primary_2
 sla_primary_1      onsoctcp   cm_host_1  cm_port_1    g=g_primary
 sla_primary_2      onsoctcp   cm_host_2  cm_port_2    g=g_primary

 g_secondaries      group      -          -            c=1,e=sla_secondaries_2
 sla_secondaries_2  onsoctcp   cm_host_1  cm_port_3    g=g_secondaries
 sla_secondaries_2  onsoctcp   cm_host_2  cm_port_4    g=g_secondaries
```

Client connection requests to **@g_primary** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_primary_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_primary_2** to provide the client application with connection information for the primary server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of high-availability clusters that use secure ports

You must define `sqlhosts` network-connectivity information for connection management of high-availability clusters. If Connection Managers, database servers, or client applications are outside of a trusted network, you must also create an encrypted password file for security.

To use a file other than `$INFORMIXDIR/etc/sqlhosts` on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and database server, create `sqlhosts` file entries for each database server in the cluster, and specify the `s=6` secure-port option. For example:

```
#dbservername   nettype    hostname   servicename   options
 server_1       onsoctcp   host_1     port_1        s=6
 server_2       onsoctcp   host_2     port_2        s=6
 server_3       onsoctcp   host_3     port_3        s=6
```

2. On the host of each Connection Manager and database server, create `sqlhosts` file alias entries for each database server. For example:

```
#dbservername   nettype    hostname   servicename   options
 server_1       onsoctcp   host_1     port_1        s=6
 a_server_1     onsoctcp   host_1     port_4

 server_2       onsoctcp   host_2     port_2        s=6
 a_server_2     onsoctcp   host_2     port_5

 server_3       onsoctcp   host_3     port_3        s=6
 a_server_3     onsoctcp   host_3     port_6
```

3. On the host of each Connection Manager, add a group entry the individual entries. Add group options to the database server and database server alias entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername   nettype    hostname   servicename   options
 my_servers     group      -          -             c=1,e=a_server_3
 server_1       onsoctcp   host_1     port_1        s=6,g=my_servers
 a_server_1     onsoctcp   host_1     port_4        g=my_servers
 server_2       onsoctcp   host_2     port_2        s=6,g=my_servers
 a_server_2     onsoctcp   host_2     port_5        g=my_servers
 server_3       onsoctcp   host_3     port_3        s=6,g=my_servers
 a_server_3     onsoctcp   host_3     port_6        g=my_servers
```

   A password file that is encrypted through the **onpassword** utility is required for connectivity through secure ports. The entries in the previously shown `sqlhosts` file are represented in the following password file.

```
my_servers  a_server_1  user_1  my_password_1
my_servers  a_server_2  user_2  my_password_2
my_servers  a_server_3  user_3  my_password_3

server_1    a_server_1  user_1  my_password_1
server_2    a_server_2  user_2  my_password_2
server_3    a_server_3  user_3  my_password_3

a_server_1  a_server_1  user_1  my_password_1
a_server_2  a_server_2  user_2  my_password_2
a_server_3  a_server_3  user_3  my_password_3
```

4. In each database server's onconfig file, set the DBSERVERALIASES parameter to that database server's alias.

   The onconfig file entry for **server_1**:

   ```
   DBSERVERALIASES a_server_1
   ```

   The onconfig file entry for **server_2**:

   ```
   DBSERVERALIASES a_server_2
   ```

   The onconfig file entry for **server_3**:

   ```
   DBSERVERALIASES a_server_3
   ```

5. On the host of each client application, create an sqlhosts file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_1

   CLUSTER my_cluster
   {
      INFORMIXSERVER my_servers
      SLA sla_primary_1     DBSERVERS=PRI
      SLA sla_secondaries_1 DBSERVERS=SDS,HDR
      FOC ORDER=ENABLED PRIORITY=1
   }
   ```

   The second Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_2

   CLUSTER my_cluster
   {
      INFORMIXSERVER my_servers
      SLA sla_primary_2     DBSERVERS=PRI
      SLA sla_secondaries_2 DBSERVERS=SDS,HDR
      FOC ORDER=ENABLED PRIORITY=2
   }
   ```

   Add the following entries to each client application's host sqlhosts file:

   ```
   #dbservername       nettype    hostname   servicename   options
    sla_primary_1      onsoctcp   cm_host_1  cm_port_1
    sla_primary_2      onsoctcp   cm_host_2  cm_port_2

    sla_secondaries_2  onsoctcp   cm_host_1  cm_port_3
    sla_secondaries_2  onsoctcp   cm_host_2  cm_port_4
   ```

6. On the host of each client application, create sqlhosts file group entries for each group of SLA entries, and add group options to the SLA entries. Use the c=1 group-entry option, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members.

   ```
   #dbservername       nettype    hostname   servicename   options
    g_primary          group      -          -             c=1,e=sla_primary_2
    sla_primary_1      onsoctcp   cm_host_1  cm_port_1     g=g_primary
    sla_primary_2      onsoctcp   cm_host_2  cm_port_2     g=g_primary
   ```

```
g_secondaries      group      -          -            c=1,e=sla_secondaries_2
sla_secondaries_2  onsoctcp   cm_host_1  cm_port_3    g=g_secondaries
sla_secondaries_2  onsoctcp   cm_host_2  cm_port_4    g=g_secondaries
```

Client connection requests to **@g_primary** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_primary_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_primary_2** to provide the client application with connection information for the primary server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for high-availability clusters that use Distributed Relational Database Architecture (DRDA)

Connection Managers support Distributed Relational Database Architecture (DRDA) connections for high-availability clusters. You must define `sqlhosts` network-connectivity information for connection management of high-availability clusters that use DRDA.

To use a file other than `$INFORMIXDIR/etc/sqlhosts` on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and database server, create `sqlhosts` file entries for each database server in the cluster. For example:

   ```
   #dbservername   nettype    hostname   servicename   options
    server_1       onsoctcp   host_1     port_1
    server_2       onsoctcp   host_3     port_2
    server_3       onsoctcp   host_5     port_3
   ```

2. In each database server's `onconfig` file, set the DBSERVERALIASES parameter to specify an alias for the server.

   The `onconfig` file entry for **server_1**:

   ```
   DBSERVERALIASES drda_1
   ```

   The `onconfig` file entry for **server_2**:

   ```
   DBSERVERALIASES drda_2
   ```

   The `onconfig` file entry for **server_3**:

   ```
   DBSERVERALIASES drda_3
   ```

3. On the host of each Connection Manager, add entries for the DRDA aliases. Use a DRDA protocol for the `nettype` value. For example:

   ```
   #dbservername   nettype    hostname   servicename   options
    server_1       onsoctcp   host_1     port_1
    server_2       onsoctcp   host_2     port_2
    server_3       onsoctcp   host_3     port_3

    drda_1         drsoctcp   host_1     port_4
    drda_2         drsoctcp   host_2     port_5
    drda_3         drsoctcp   host_3     port_6
   ```

4. On the host of each Connection Manager, add a group entry for the group of database server and add a group entry for the group of DRDA aliases. Add group options to the database server and DRDA alias entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername   nettype    hostname   servicename   options
 my_servers     group      -          -             c=1,e=server_3
 server_1       onsoctcp   host_1     port_1        g=my_servers
 server_2       onsoctcp   host_2     port_2        g=my_servers
 server_3       onsoctcp   host_3     port_3        g=my_servers

 drda_aliases   group      -          -             c=1,e=drda_3
 drda_1         drsoctcp   host_1     port_4        g=drda_aliases
 drda_2         drsoctcp   host_2     port_5        g=drda_aliases
 drda_3         drsoctcp   host_3     port_6        g=drda_aliases
```

5. On the host of each client application, create an `sqlhosts` file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

```
NAME connection_manager_1

CLUSTER my_cluster
{
   INFORMIXSERVER my_servers
   SLA sla_primary_1          DBSERVERS=PRI
   SLA sla_primary_drda_1     DBSERVERS=PRI
   SLA sla_secondaries_1      DBSERVERS=SDS,HDR
   SLA sla_secondaries_drda_1 DBSERVERS=SDS,HDR
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

   The second Connection Manager's configuration file has the following entries:

```
NAME connection_manager_2

CLUSTER my_cluster
{
   INFORMIXSERVER my_servers
   SLA sla_primary_2          DBSERVERS=PRI
   SLA sla_primary_drda_2     DBSERVERS=PRI
   SLA sla_secondaries_2      DBSERVERS=SDS,HDR
   SLA sla_secondaries_drda_2 DBSERVERS=SDS,HDR
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

   Add the following entries to each client application's host `sqlhosts` file:

```
#dbservername            nettype    hostname   servicename  options
 sla_primary_1           onsoctcp   cm_host_1  cm_port_1
 sla_primary_2           onsoctcp   cm_host_2  cm_port_2

 sla_secondaries_2       onsoctcp   cm_host_1  cm_port_3
 sla_secondaries_2       onsoctcp   cm_host_2  cm_port_4

 sla_primary_1_drda      drsoctcp   cm_host_1  cm_port_5
 sla_primary_2_drda      drsoctcp   cm_host_2  cm_port_6

 sla_secondaries_2_drda  drsoctcp   cm_host_1  cm_port_7
 sla_secondaries_2_drda  drsoctcp   cm_host_2  cm_port_8
```

6. On the host of each client application, create `sqlhosts` file group entries for each group of SLA entries, and add group options to the SLA entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members.

```
#dbservername            nettype    hostname   servicename  options
 g_primary               group      -          -            c=1,e=sla_primary_2
 sla_primary_1           onsoctcp   cm_host_1  cm_port_1    g=g_primary
 sla_primary_2           onsoctcp   cm_host_2  cm_port_2    g=g_primary

 g_secondaries           group      -          -            c=1,e=sla_secondaries_2
```

```
sla_secondaries_2       onsoctcp   cm_host_1  cm_port_3    g=g_secondaries
sla_secondaries_2       onsoctcp   cm_host_2  cm_port_4    g=g_secondaries

g_primary_drda          group      -          -            c=1,e=sla_primary_2_drda
sla_primary_1_drda      drsoctcp   cm_host_1  cm_port_5    g=g_primary_drda
sla_primary_2_drda      drsoctcp   cm_host_2  cm_port_6    g=g_primary_drda

g_secondaries_drda      group      -          -            c=1,e=sla_secondaries_2_drda
sla_secondaries_2_drda  drsoctcp   cm_host_1  cm_port_7    g=g_secondaries_drda
sla_secondaries_2_drda  drsoctcp   cm_host_2  cm_port_8    g=g_secondaries_drda
```

Client connection requests to **@g_primary_drda** are sent by **drsoctcp** protocol to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_primary_1_drda** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_primary_2_drda** to provide the client application with connection information for the primary server.

Related concepts:

"The sqlhosts information" on page 2-18

Related tasks:

"Configuring connectivity between Informix database servers and IBM Data Server clients" on page 2-43

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for high-availability clusters that use Distributed Relational Database Architecture (DRDA) and secure ports

Connection Managers support Distributed Relational Database Architecture (DRDA) connections for high-availability clusters. You must define `sqlhosts` network-connectivity information for connection management of high-availability clusters that use DRDA. If Connection Managers, database servers, or client applications are outside of a trusted network, you must also create an encrypted password file for security.

The Connection Manager's `sqlhosts` file must contain entries for all database servers that it connects to.

To use a file other than `$INFORMIXDIR/etc/sqlhosts` on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and database server, create `sqlhosts` file entries for each database server in the cluster, and specify the `s=6` secure-port option. For example:

   ```
   #dbservername  nettype   hostname  servicename  options
    server_1      onsoctcp  host_1    port_1       s=6
    server_2      onsoctcp  host_3    port_2       s=6
    server_3      onsoctcp  host_5    port_3       s=6
   ```

2. On the host of each Connection Manager and database server, add DRDA alias entries. Use a DRDA protocol for the `nettype` value, and specify the `s=6` secure-port option. For example:

   ```
   #dbservername  nettype   hostname  servicename  options
    server_1      onsoctcp  host_1    port_1       s=6
    server_2      onsoctcp  host_2    port_2       s=6
    server_3      onsoctcp  host_3    port_3       s=6
   ```

```
drda_1        drsoctcp  host_1    port_4       s=6
drda_2        drsoctcp  host_2    port_5       s=6
drda_3        drsoctcp  host_3    port_6       s=6
```

3. On the host of each Connection Manager and database server, create `sqlhosts` file alias entries for each database server and each DRDA alias. For example:

```
#dbservername  nettype   hostname  servicename  options
 server_1      onsoctcp  host_1    port_1       s=6
 a_server_1    onsoctcp  host_1    port_7

 server_2      onsoctcp  host_2    port_2       s=6
 a_server_2    onsoctcp  host_2    port_8

 server_3      onsoctcp  host_3    port_3       s=6
 a_server_3    onsoctcp  host_3    port_9

 drda_1        drsoctcp  host_1    port_4       s=6
 a_drda_1      drsoctcp  host_1    port_10

 drda_2        drsoctcp  host_2    port_5       s=6
 a_drda_2      drsoctcp  host_2    port_11

 drda_3        drsoctcp  host_3    port_6       s=6
 a_drda_3      drsoctcp  host_3    port_12
```

4. On the host of each Connection Manager, add group entries for the groups of database servers and DRDA entries. Add group options to the individual entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername  nettype   hostname  servicename  options
 g_servers     group     -         -            c=1,e=a_server_3
 server_1      onsoctcp  host_1    port_1       s=6,g=g_servers
 a_server_1    onsoctcp  host_1    port_7       g=g_servers
 server_2      onsoctcp  host_2    port_2       s=6,g=g_servers
 a_server_2    onsoctcp  host_2    port_8       g=g_servers
 server_3      onsoctcp  host_3    port_3       s=6,g=g_servers
 a_server_3    onsoctcp  host_3    port_9       g=g_servers

 g_drda        group     -         -            c=1,e=a_drda_3
 drda_1        drsoctcp  host_1    port_4       s=6,g=g_drda
 a_drda_1      drsoctcp  host_1    port_10      g=g_drda
 drda_2        drsoctcp  host_2    port_5       s=6,g=g_drda
 a_drda_2      drsoctcp  host_2    port_11      g=g_drda
 drda_3        drsoctcp  host_3    port_6       s=6,g=g_drda
 a_drda_3      drsoctcp  host_3    port_12      g=g_drda
```

A password file that is encrypted through the **onpassword** utility is required for connectivity through secure ports. The entries in the previously shown `sqlhosts` file are represented in the following password file.

```
g_servers   a_server_1  user_1  password_1
g_servers   a_server_2  user_2  password_2
g_servers   a_server_3  user_3  password_3

server_1    a_server_1  user_1  password_1
server_2    a_server_2  user_2  password_2
server_3    a_server_3  user_3  password_3

a_server_1  a_server_1  user_1  password_1
a_server_2  a_server_2  user_2  password_2
a_server_3  a_server_3  user_3  password_3

g_drda      a_drda_1    user_1  password_1
g_drda      a_drda_2    user_2  password_2
```

```
g_drda      a_drda_3    user_3  password_3

drda_1      a_drda_1    user_1  password_1
drda_2      a_drda_2    user_2  password_2
drda_3      a_drda_3    user_3  password_3

a_drda_1    a_drda_1    user_1  password_1
a_drda_2    a_drda_2    user_2  password_2
a_drda_3    a_drda_3    user_3  password_3
```

5. In each database server's onconfig file, set the DBSERVERALIASES parameter to that database server's aliases.

   The onconfig file entry for **server_1**:

   ```
   DBSERVERALIASES a_server_1,drda_1,a_drda_1
   ```

   The onconfig file entry for **server_2**:

   ```
   DBSERVERALIASES a_server_2,drda_2,a_drda_2
   ```

   The onconfig file entry for **server_3**:

   ```
   DBSERVERALIASES a_server_3,drda_3,a_drda_3
   ```

6. On the host of each client application, create an sqlhosts file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_1

   CLUSTER my_cluster
   {
      INFORMIXSERVER g_servers,g_drda
      SLA sla_primary_1          DBSERVERS=PRI
      SLA sla_primary_drda_1     DBSERVERS=PRI
      SLA sla_secondaries_1      DBSERVERS=SDS,HDR
      SLA sla_secondaries_drda_1 DBSERVERS=SDS,HDR
      FOC ORDER=ENABLED \
          PRIORITY=1
   }
   ```

   The second Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_2

   CLUSTER my_cluster
   {
      INFORMIXSERVER g_servers,g_drda
      SLA sla_primary_2          DBSERVERS=PRI
      SLA sla_primary_drda_2     DBSERVERS=PRI
      SLA sla_secondaries_2      DBSERVERS=SDS,HDR
      SLA sla_secondaries_drda_2 DBSERVERS=SDS,HDR
      FOC ORDER=ENABLED \
          PRIORITY=2
   }
   ```

   Add the following entries to each client application's host sqlhosts file:

   ```
   #dbservername            nettype   hostname   servicename  options
    sla_primary_1           onsoctcp  cm_host_1  cm_port_1
    sla_primary_2           onsoctcp  cm_host_2  cm_port_2

    sla_secondaries_2       onsoctcp  cm_host_1  cm_port_3
    sla_secondaries_2       onsoctcp  cm_host_2  cm_port_4

    sla_primary_1_drda      drsoctcp  cm_host_1  cm_port_5
    sla_primary_2_drda      drsoctcp  cm_host_2  cm_port_6

    sla_secondaries_2_drda  drsoctcp  cm_host_1  cm_port_7
    sla_secondaries_2_drda  drsoctcp  cm_host_2  cm_port_8
   ```

7. On the host of each client application, create sqlhosts file group entries for each group of SLA entries, and add group options to the SLA entries. Use the

c=1 group-entry option, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members.

```
#dbservername            nettype    hostname   servicename  options
 g_primary               group      -          -            c=1,e=sla_primary_2
 sla_primary_1           onsoctcp   cm_host_1  cm_port_1    g=g_primary
 sla_primary_2           onsoctcp   cm_host_2  cm_port_2    g=g_primary

 g_secondaries           group      -          -            c=1,e=sla_secondaries_2
 sla_secondaries_2       onsoctcp   cm_host_1  cm_port_3    g=g_secondaries
 sla_secondaries_2       onsoctcp   cm_host_2  cm_port_4    g=g_secondaries

 g_primary_drda          group      -          -            c=1,e=sla_primary_2_drda
 sla_primary_1_drda      drsoctcp   cm_host_1  cm_port_5    g=g_primary_drda
 sla_primary_2_drda      drsoctcp   cm_host_2  cm_port_6    g=g_primary_drda

 g_secondaries_drda      group      -          -            c=1,e=sla_secondaries_2_drda
 sla_secondaries_2_drda  drsoctcp   cm_host_1  cm_port_7    g=g_secondaries_drda
 sla_secondaries_2_drda  drsoctcp   cm_host_2  cm_port_8    g=g_secondaries_drda
```

Client connection requests to **@g_primary_drda** are sent by **drsoctcp** protocol to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_primary_1_drda** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_primary_2_drda** to provide the client application with connection information for the primary server.

Related concepts:

"The sqlhosts information" on page 2-18

Related tasks:

"Configuring connectivity between Informix database servers and IBM Data Server clients" on page 2-43

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of grids and replicate sets

You must define sqlhosts network-connectivity information for connection management of replicate sets or grids.

To use a file other than $INFORMIXDIR/etc/sqlhosts on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and replication server, create sqlhosts file entries for each replication server. For example:

   ```
   #dbservername  nettype   hostname   servicename  options
    server_1      onsoctcp  host_1     port_1
    server_2      onsoctcp  host_2     port_2
    server_3      onsoctcp  host_3     port_3
    server_4      onsoctcp  host_4     port_4
   ```

2. On the host of each Connection Manager and replication server, create a sqlhosts file group entry for each replication server. Add group options to each replication-server entry. Use the i=*unique_number* group-entry option to assign an identifier to the group for Enterprise Replication. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

   ```
   #dbservername  nettype   hostname   servicename  options
    g_server_1    group     -          -            i=1,e=server_1
    server_1      onsoctcp  host_1     port_1       g=g_server_1
   ```

```
g_server_2    group     -          -           i=2,e=server_2
server_2      onsoctcp  host_2     port_2      g=g_server_2

g_server_3    group     -          -           i=3,e=server_3
server_3      onsoctcp  host_3     port_3      g=g_server_3

g_server_4    group     -          -           i=4,e=server_4
server_4      onsoctcp  host_4     port_4      g=g_server_4
```

3. On the host of each client application, create an `sqlhosts` file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_1

   REPLSET my_replset
   {
      INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
      SLA sla_1 DBSERVERS=ANY
   }
   ```

   The second Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_2

   REPLSET my_replset
   {
      INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
      SLA sla_2 DBSERVERS=ANY
   }
   ```

   Add the following entries to each client application's host `sqlhosts` file:

   ```
   #dbservername  nettype   hostname   servicename  options
    sla_1         onsoctcp  cm_host_1  cm_port_1
    sla_2         onsoctcp  cm_host_2  cm_port_2
   ```

4. On the host of each client application, create `sqlhosts` file group entries for each group of SLA entries, and add group options to the SLA entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

   ```
   #dbservername  nettype   hostname   servicename  options
    g_sla         onsoctcp  -          -             c=1,e=sla_2
    sla_1         onsoctcp  cm_host_1  cm_port_1     g=g_sla
    sla_2         onsoctcp  cm_host_2  cm_port_2     g=g_sla
   ```

Client connection requests to **@g_sla** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_2** to provide the client application with connection information for a replication server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of grids and replicate sets that use secure ports

You must define `sqlhosts` network-connectivity information for connection management of replicate sets or grids. If Connection Managers, database servers, or client applications are outside of a trusted network, you must also create an encrypted password file for security.

To use a file other than $INFORMIXDIR/etc/sqlhosts on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and replication server, create sqlhosts file entries for each replication server, and specify the s=6 secure-port option. For example:

```
#dbservername  nettype   hostname   servicename  options
 server_1      onsoctcp  host_1     port_1       s=6
 server_2      onsoctcp  host_2     port_3       s=6
 server_3      onsoctcp  host_3     port_5       s=6
 server_4      onsoctcp  host_4     port_7       s=6
```

2. On the host of each Connection Manager and replication server, create a sqlhosts file alias entry for each replication server. For example:

```
#dbservername  nettype   hostname  servicename  options
 server_1      onsoctcp  host_1    port_1       s=6
 a_server_1    onsoctcp  host_1    port_2

 server_2      onsoctcp  host_2    port_3       s=6
 a_server_2    onsoctcp  host_2    port_4

 server_3      onsoctcp  host_3    port_5       s=6
 a_server_3    onsoctcp  host_3    port_6

 server_4      onsoctcp  host_4    port_7       s=6
 a_server_4    onsoctcp  host_4    port_8
```

   The aliases are used by the **cdr utility**, which cannot connect to a secure port.

3. On the host of each Connection Manager and replication server, create a sqlhosts file group entry for each replication server and alias pair. Use the i=*unique_number* group-entry option to assign an identifier to the group for Enterprise Replication. Add group options to each replication server and alias entry. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

```
#dbservername  nettype   hostname   servicename  options
 g_server_1    group     -          -            i=1,e=a_server_1
 server_1      onsoctcp  host_1     port_1       g=g_server_1,s=6
 a_server_1    onsoctcp  host_1     port_2       g=g_server_1

 g_server_2    group     -          -            i=2,e=a_server_2
 server_2      onsoctcp  host_2     port_3       g=g_server_2,s=6
 a_server_2    onsoctcp  host_2     port_4       g=g_server_2

 g_server_3    group     -          -            i=3,e=a_server_3
 server_3      onsoctcp  host_3     port_5       g=g_server_3,s=6
 a_server_3    onsoctcp  host_3     port_6       g=g_server_3

 g_server_4    group     -          -            i=4,e=a_server_4
 server_4      onsoctcp  host_4     port_7       g=g_server_4,s=6
 a_server_4    onsoctcp  host_4     port_8       g=g_server_4
```

   A password file that is encrypted through the **onpassword** utility is required for connectivity through secure ports. The entries in the previously shown sqlhosts file are represented in the following password file.

```
g_server_1  a_server_1  user_1  my_password_1
server_1    a_server_1  user_1  my_password_1
a_server_1  a_server_1  user_1  my_password_1

g_server_2  a_server_2  user_2  my_password_2
server_2    a_server_2  user_2  my_password_2
a_server_2  a_server_2  user_2  my_password_2

g_server_3  a_server_3  user_3  my_password_3
server_3    a_server_3  user_3  my_password_3
```

```
 a_server_3  a_server_3  user_3  my_password_3

 g_server_4  a_server_4  user_4  my_password_4
 server_4    a_server_4  user_4  my_password_4
 a_server_4  a_server_4  user_4  my_password_4
```

4. In each replication server's onconfig file, set the DBSERVERALIASES parameter to that database server's aliases.

   The onconfig file entry for **server_1**:

   ```
   DBSERVERALIASES a_server_1
   ```

   The onconfig file entry for **server_2**:

   ```
   DBSERVERALIASES a_server_2
   ```

   The onconfig file entry for **server_3**:

   ```
   DBSERVERALIASES a_server_3
   ```

   The onconfig file entry for **server_4**:

   ```
   DBSERVERALIASES a_server_4
   ```

5. On the host of each client application, create an sqlhosts file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_1

   REPLSET my_replset
   {
      INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
      SLA sla_1 DBSERVERS=ANY
   }
   ```

   The second Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_2

   REPLSET my_replset
   {
      INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
      SLA sla_2 DBSERVERS=ANY
   }
   ```

   Add the following entries to each client application's host sqlhosts file:

   ```
   #dbservername  nettype   hostname   servicename  options
    sla_1         onsoctcp  cm_host_1  cm_port_1
    sla_2         onsoctcp  cm_host_2  cm_port_2
   ```

6. On the host of each client application, create sqlhosts file group entries for each group of SLA entries, and add group options to the SLA entries. Use the c=1 group-entry option, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

   ```
   #dbservername  nettype   hostname   servicename  options
    g_sla         group     -          -            c=1,e=sla_2
    sla_1         onsoctcp  cm_host_1  cm_port_1    g=g_sla
    sla_2         onsoctcp  cm_host_2  cm_port_2    g=g_sla
   ```

Client connection requests to **@g_sla** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_2** to provide the client application with connection information for the primary server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management high-availability replication systems

You must define sqlhosts network-connectivity information for connection management of high-availability replication systems.

To use a file other than $INFORMIXDIR/etc/sqlhosts on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

For this example, you are setting up Enterprise Replication between the primary servers of three high-availability clusters.

Cluster 1:

- server_1 (primary)
- server_2 (SD secondary)
- server_3 (HDR secondary)
- server_4 (RS secondary)

Cluster 2:

- server_5 (primary)
- server_6 (SD secondary)
- server_7 (HDR secondary)
- server_8 (RS secondary)

Cluster 3:

- server_9 (primary)
- server_10 (SD secondary)
- server_11 (HDR secondary)
- server_12 (RS secondary)

1. On the host of each Connection Manager and database server, create sqlhosts file entries for each server. For example:

```
#dbservername  nettype   hostname   servicename  options
server_1       onsoctcp  host_1     port_1
server_2       onsoctcp  host_1     port_2
server_3       onsoctcp  host_2     port_3
server_4       onsoctcp  host_3     port_4

server_5       onsoctcp  host_4     port_5
server_6       onsoctcp  host_4     port_6
server_7       onsoctcp  host_5     port_7
server_8       onsoctcp  host_6     port_8

server_9       onsoctcp  host_7     port_9
server_10      onsoctcp  host_7     port_10
server_11      onsoctcp  host_8     port_11
server_12      onsoctcp  host_9     port_12
```

2. On the host of each Connection Manager and database server, create a sqlhosts file group entry for each replication-server entry and each cluster. Add group options to each database server entry. Use the i=*unique_number* group-entry option to assign an identifier to the group for Enterprise Replication. Use the c=1 group-entry option for cluster groups, so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

```
#dbservername  nettype   hostname   servicename  options
 cluster_1     group     -          -            i=1,c=1,e=server_4
 server_1      onsoctcp  host_1     port_1       g=cluster_1
 server_2      onsoctcp  host_1     port_2       g=cluster_1
 server_3      onsoctcp  host_2     port_3       g=cluster_1
 server_4      onsoctcp  host_3     port_4       g=cluster_1

 cluster_2     group     -          -            i=2,c=1,e=server_8
 server_5      onsoctcp  host_4     port_5       g=cluster_2
 server_6      onsoctcp  host_4     port_6       g=cluster_2
 server_7      onsoctcp  host_5     port_7       g=cluster_2
 server_8      onsoctcp  host_6     port_8       g=cluster_2

 cluster_3     group     -          -            i=3,c=1,e=server_12
 server_9      onsoctcp  host_7     port_9       g=cluster_3
 server_10     onsoctcp  host_7     port_10      g=cluster_3
 server_11     onsoctcp  host_8     port_11      g=cluster_3
 server_12     onsoctcp  host_9     port_12      g=cluster_3
```

3. On the host of each client application, create an `sqlhosts` file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

```
NAME connection_manager_1

REPLSET my_replset
{
   INFORMIXSERVER cluster_1,cluster_2,cluster_3
   SLA sla_1 DBSERVERS=ANY
}

CLUSTER my_cluster_1
{
   INFORMIXSERVER cluster_1
   FOC ORDER=ENABLED \
       PRIORITY=1
}

CLUSTER my_cluster_2
{
   INFORMIXSERVER cluster_2
   FOC ORDER=ENABLED \
       PRIORITY=1
}

CLUSTER my_cluster_3
{
   INFORMIXSERVER cluster_3
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

   The second Connection Manager's configuration file has the following entries:

```
NAME connection_manager_2

REPLSET my_replset
{
   INFORMIXSERVER cluster_1,cluster_2,cluster_3
   SLA sla_2 DBSERVERS=ANY
}

CLUSTER my_cluster_1
{
   INFORMIXSERVER cluster_1
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

```
CLUSTER my_cluster_2
{
   INFORMIXSERVER cluster_2
   FOC ORDER=ENABLED \
       PRIORITY=2
}

CLUSTER my_cluster_3
{
   INFORMIXSERVER cluster_3
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

Add the following entries to each client application's host `sqlhosts` file:

```
#dbservername  nettype   hostname   servicename  options
 sla_1         onsoctcp  cm_host_1  cm_port_1
 sla_2         onsoctcp  cm_host_2  cm_port_2
```

4. On the host of each client application, create `sqlhosts` file group entries for each group of SLA entries, and add group options to the SLA entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername  nettype   hostname   servicename  options
 g_sla         onsoctcp  -          -            c=1,e=sla_2
 sla_1         onsoctcp  cm_host_1  cm_port_1    g=g_sla
 sla_2         onsoctcp  cm_host_2  cm_port_2    g=g_sla
```

Client connection requests to **@g_sla** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_2** to provide the client application with connection information for a replication server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of high-availability replication systems that use secure ports

You must define `sqlhosts` network-connectivity information for connection management of high-availability replication systems. If Connection Managers, database servers, or client applications are outside of a trusted network, you must also create an encrypted password file for security.

To use a file other than `$INFORMIXDIR/etc/sqlhosts` on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

For this example, you are setting up Enterprise Replication between the primary servers of two high-availability clusters.

Cluster 1:

- **server_1** (primary)
- **server_2** (SD secondary)
- **server_3** (HDR secondary)
- **server_4** (RS secondary)

Cluster 2:

- **server_5** (primary)
- **server_6** (SD secondary)
- **server_7** (HDR secondary)
- **server_8** (RS secondary)

1. On the host of each Connection Manager and database server, create `sqlhosts` file entries for each database server. For example:

```
#dbservername  nettype   hostname  servicename  options
 server_1      onsoctcp  host_1    port_1       s=6
 server_2      onsoctcp  host_1    port_2       s=6
 server_3      onsoctcp  host_2    port_3       s=6
 server_4      onsoctcp  host_3    port_4       s=6

 server_5      onsoctcp  host_4    port_5       s=6
 server_6      onsoctcp  host_4    port_6       s=6
 server_7      onsoctcp  host_5    port_7       s=6
 server_8      onsoctcp  host_6    port_8       s=6
```

2. On the host of each Connection Manager and database server, create a `sqlhosts` file alias entry for each database server. For example:

```
#dbservername  nettype   hostname  servicename  options
 server_1      onsoctcp  host_1    port_1       s=6
 a_server_1    onsoctcp  host_1    port_9

 server_2      onsoctcp  host_1    port_2       s=6
 a_server_2    onsoctcp  host_1    port_10

 server_3      onsoctcp  host_2    port_3       s=6
 a_server_3    onsoctcp  host_2    port_11

 server_4      onsoctcp  host_3    port_4       s=6
 a_server_4    onsoctcp  host_3    port_12

 server_5      onsoctcp  host_4    port_5       s=6
 a_server_5    onsoctcp  host_4    port_13

 server_6      onsoctcp  host_4    port_6       s=6
 a_server_6    onsoctcp  host_4    port_14

 server_7      onsoctcp  host_5    port_7       s=6
 a_server_7    onsoctcp  host_5    port_15

 server_8      onsoctcp  host_6    port_8       s=6
 a_server_8    onsoctcp  host_6    port_16
```

   The aliases are used by the **cdr utility**, which cannot connect to a secure port.

3. On the host of each Connection Manager and database server, create a `sqlhosts` file group entry for each cluster. Add group options to each database server entry. Use the `i=`*`unique_number`* group-entry option to assign an identifier to the group for Enterprise Replication. Use the `c=1` group-entry option for cluster groups, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername  nettype   hostname  servicename  options
 cluster_1     group     -          -           i=1,c=1,e=a_server_4
 server_1      onsoctcp  host_1    port_1       s=6,g=cluster_1
 a_server_1    onsoctcp  host_1    port_9       g=cluster_1
 server_2      onsoctcp  host_1    port_2       s=6,g=cluster_1
 a_server_2    onsoctcp  host_1    port_10      g=cluster_1
 server_3      onsoctcp  host_2    port_3       s=6,g=cluster_1
 a_server_3    onsoctcp  host_2    port_11      g=cluster_1
```

```
server_4      onsoctcp  host_3    port_4       s=6,g=cluster_1
a_server_4    onsoctcp  host_3    port_12      g=cluster_1

cluster_2     group     -          -           i=1,c=1,e=a_server_8
server_5      onsoctcp  host_4    port_5       s=6,g=cluster_2
a_server_5    onsoctcp  host_4    port_13      g=cluster_2
server_6      onsoctcp  host_4    port_6       s=6,g=cluster_2
a_server_6    onsoctcp  host_4    port_14      g=cluster_2
server_7      onsoctcp  host_5    port_7       s=6,g=cluster_2
a_server_7    onsoctcp  host_5    port_15      g=cluster_2
server_8      onsoctcp  host_6    port_8       s=6,g=cluster_2
a_server_8    onsoctcp  host_6    port_16      g=cluster_2
```

A password file that is encrypted through the **onpassword** utility is required for connectivity through secure ports. The entries in the previously shown `sqlhosts` file are represented in the following password file.

```
cluster_1   a_server_1  user_1  password_1
cluster_1   a_server_2  user_2  password_2
cluster_1   a_server_3  user_3  password_3
cluster_1   a_server_4  user_4  password_4

cluster_2   a_server_5  user_5  password_5
cluster_2   a_server_6  user_6  password_6
cluster_2   a_server_7  user_7  password_7
cluster_2   a_server_8  user_8  password_8

server_1    a_server_1  user_1  password_1
server_2    a_server_2  user_2  password_2
server_3    a_server_3  user_3  password_3
server_4    a_server_4  user_4  password_4
server_5    a_server_5  user_5  password_5
server_6    a_server_6  user_6  password_6
server_7    a_server_7  user_7  password_7
server_8    a_server_8  user_8  password_8

a_server_1  a_server_1  user_1  password_1
a_server_2  a_server_2  user_2  password_2
a_server_3  a_server_3  user_3  password_3
a_server_4  a_server_4  user_4  password_4
a_server_5  a_server_5  user_5  password_5
a_server_6  a_server_6  user_6  password_6
a_server_7  a_server_7  user_7  password_7
a_server_8  a_server_8  user_8  password_8
```

4. In each database server's `onconfig` file, set the DBSERVERALIASES parameter to that database server's aliases. For example:

   The `onconfig` file entry for **server_1**:

   ```
   DBSERVERALIASES a_server_1
   ```

5. On the host of each client application, create an `sqlhosts` file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

   ```
   NAME connection_manager_1

   REPLSET my_replset
   {
      INFORMIXSERVER cluster_1,cluster_2
      SLA sla_1 DBSERVERS=ANY
   }

   CLUSTER my_cluster_1
   {
      INFORMIXSERVER cluster_1
      FOC ORDER=ENABLED \
          PRIORITY=1
   ```

```
}

CLUSTER my_cluster_2
{
   INFORMIXSERVER cluster_2
   FOC ORDER=ENABLED \
       PRIORITY=1
}
```

The second Connection Manager's configuration file has the following entries:

```
NAME connection_manager_2

REPLSET my_replset
{
   INFORMIXSERVER cluster_1,cluster_2
   SLA sla_2 DBSERVERS=ANY
}

CLUSTER my_cluster_1
{
   INFORMIXSERVER cluster_1
   FOC ORDER=ENABLED \
       PRIORITY=2
}

CLUSTER my_cluster_2
{
   INFORMIXSERVER cluster_2
   FOC ORDER=ENABLED \
       PRIORITY=2
}
```

Add the following entries to each client application's host `sqlhosts` file:

```
#dbservername  nettype   hostname   servicename  options
 sla_1         onsoctcp  cm_host_1  cm_port_1
 sla_2         onsoctcp  cm_host_2  cm_port_2
```

6. On the host of each client application, create `sqlhosts` file group entries for each group of SLA entries, and add group options to the SLA entries. Use the `c=1` group-entry option, so that connection-attempt starting points in the list of group members is random. Use the `e=`*`last_member`* group-entry option so that the entire `sqlhosts` is not scanned for group members. For example:

```
#dbservername  nettype   hostname   servicename  options
 g_sla         onsoctcp  -          -             c=1,e=sla_2
 sla_1         onsoctcp  cm_host_1  cm_port_1     g=g_sla
 sla_2         onsoctcp  cm_host_2  cm_port_2     g=g_sla
```

Client connection requests to **@g_sla** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_1** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_2** to provide the client application with connection information for a replication server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Defining sqlhosts information for connection management of server sets

You must define `sqlhosts` network-connectivity information for connection management of server sets.

The Connection Manager's sqlhosts file must contain entries for all database servers that it connects to.

To use a file other than $INFORMIXDIR/etc/sqlhosts on a specific host, set the host's **INFORMIXSQLHOSTS** environment variable to the alternative file.

1. On the host of each Connection Manager and database server, create sqlhosts file entries for each database server. For example:

```
#dbservername   nettype    hostname   servicename   options
 standalone_1   onsoctcp   host_1     port_1

 standalone_2   onsoctcp   host_2     port_2

 standalone_3   onsoctcp   host_3     port_3
```

2. On the host of each client application, create an sqlhosts file entry for each service-level agreement (SLA) in each Connection Manager configuration file.

   The first Connection Manager's configuration file has the following entries:

```
NAME connection_manager_1
MACRO servers=standalone_1,standalone_2,standalone_3

SERVERSET ${servers}
{
   INFORMIXSERVER ${servers}
   SLA sla_1a DBSERVERS=standalone_1
   SLA sla_2a DBSERVERS=standalone_2
   SLA sla_3a DBSERVERS=standalone_3
}
```

   The second Connection Manager's configuration file has the following entries:

```
NAME connection_manager_2
MACRO servers=standalone_1,standalone_2,standalone_3

SERVERSET ${servers}
{
   INFORMIXSERVER ${servers}
   SLA sla_1b DBSERVERS=standalone_1
   SLA sla_2b DBSERVERS=standalone_2
   SLA sla_3b DBSERVERS=standalone_3
}
```

   Add the following entries to each client application's host sqlhosts file:

```
#dbservername  nettype   hostname   servicename  options
 sla_1a        onsoctcp  cm_host_1  cm_port_1
 sla_1b        onsoctcp  cm_host_2  cm_port_2

 sla_2a        onsoctcp  cm_host_1  cm_port_3
 sla_2b        onsoctcp  cm_host_2  cm_port_4

 sla_3a        onsoctcp  cm_host_1  cm_port_5
 sla_3b        onsoctcp  cm_host_2  cm_port_6
```

3. On the host of each client application, create sqlhosts file group entries for each group of SLA entries, and add group options to the SLA entries. Use the c=1 group-entry option so that connection-attempt starting points in the list of group members is random. Use the e=*last_member* group-entry option so that the entire sqlhosts is not scanned for group members. For example:

```
#dbservername  nettype   hostname   servicename  options
 sla_1         group     -          -            c=1,e=sla_1b
 sla_1a        onsoctcp  cm_host_1  cm_port_1    g=sla_1
 sla_1b        onsoctcp  cm_host_2  cm_port_2    g=sla_1

 sla_2         group     -          -            c=1,e=sla_2b
 sla_2a        onsoctcp  cm_host_1  cm_port_3    g=sla_2
 sla_2b        onsoctcp  cm_host_2  cm_port_4    g=sla_2
```

```
sla_3         group     -          -           c=1,e=sla_3b
sla_3a        onsoctcp  cm_host_1  cm_port_5   g=sla_3
sla_3b        onsoctcp  cm_host_2  cm_port_6   g=sla_3
```

Client connection requests to **@sla_1** are directed to one of the Connection Managers. If **connection_manager_1** receives the request, it uses **sla_1a** to provide the client application with connection information for the primary server. If **connection_manager_2** receives the request, it uses **sla_1b** to provide the client application with connection information for a replication server.

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:

"Group information" on page 2-31

Creating a password file for connecting to database servers on untrusted networks

If a client, Connection Manager, or any of the database servers that a Connection Manager connects to are on an untrusted network, you can create encrypted password files to verify connection requests.

In certain situations, an encrypted password file is required for trusted network environments, such as when a local system account attempts to connect to a database server in a high-availability cluster or Enterprise Replication domain, or when the user ID does not exist on a database server. The password file provides the correct system-level access, so that a local system account or a Windows account can connect directly to a remote server.

The password file has separate entries for the following items:

- Each Enterprise Replication group
- Each High-availability cluster group
- Each High-availability cluster server
- Each Enterprise Replication server that is in a group that is also configured for high-availability
- Each database server's alternative server alias, if the database server is using a secure port for communication

A password file entry contains the following information:

- The name of an alternative server to connect to if a connection cannot be made to the listed server or group. For example, *alternative_server_name* is used when *server_or_group_name* uses a secure port, as specified by the `s=6` option in an `sqlhosts` file entry.
- The user ID for a database server or the database servers in a group. User IDs must have the following privileges:
  - Permission to connect to the **sysadmin** database
  - CONNECT permission on the remote servers
  - On UNIX operating systems, membership in the group **informix** DBSA group
  - On Windows operating systems, membership in the **Informix-Admin** DBSA group

  Only user **informix** has all of these privileges by default
- The password for a server

1. On a Connection Manager host, use a text editor to create an ASCII text file to be used as a password file. Save the file to the $INFORMIXDIR/tmp directory. If you have a high-availability replication system, your password file contains password information for replication servers and cluster servers.

   Note: The password file must not contain comments.

   The replication-server entries of the password file have the following format:

```
group_name             database_server_alias  user_name  database_server_password
database_server_name   database_server_alias  user_name  database_server_password
database_server_alias  database_server_alias  user_name  database_server_password
```

   For example:

```
group_1   unsecure_server_alias_1  user_1  password_1
server_1  unsecure_server_alias_1  user_1  password_1
alias_1   unsecure_server_alias_1  user_1  password_1

group_2   unsecure_server_alias_2  user_2  password_2
server_2  unsecure_server_alias_2  user_2  password_2
alias_2   unsecure_server_alias_2  user_2  password_2

group_n   unsecure_server_alias_n  user_n  password_n
server_n  unsecure_server_alias_n  user_n  password_n
alias_n   unsecure_server_alias_n  user_n  password_n
```

   The cluster-server entries of the password file have the following format:

```
alias_group_name  db_server_alias  user_name  db_server_password

db_server_name    db_server_alias  user_name  db_server_password
```

   For example:

```
alias_group_1  unsecure_alias_1  user_1  password_1
alias_group_1  unsecure_alias_2  user_2  password_2
alias_group_1  unsecure_alias_n  user_n  password_n

alias_group_2  unsecure_alias_1  user_1  password_1
alias_group_2  unsecure_alias_2  user_2  password_2
alias_group_2  unsecure_alias_n  user_n  password_n

alias_group_n  unsecure_alias_1  user_1  password_1
alias_group_n  unsecure_alias_2  user_2  password_2
alias_group_n  unsecure_alias_n  user_n  password_n

server_1       unsecure_alias_1  user_1  password_1
server_2       unsecure_alias_2  user_2  password_2
server_n       unsecure_alias_n  user_n  password_n
```

2. Encrypt the password file with the **onpassword** utility and an encryption key. For example, if your password file is $INFORMIXDIR/tmp/my_passwords.txt, and the encryption key you want to use is **my_secret_encryption_key_efgh**, run the following command:

   ```
   onpassword -k my_secret_encryption_key_efgh -e my_passwords.txt
   ```

   This example creates the encrypted passwd_file file in the $INFORMIXDIR/etc directory. To later decrypt the password file, you must enter the same key that was used to encrypt the password file. If you lose the encryption key that was used to encrypt a password file, re-encrypt the original ASCII text password file. If the ASCII text password file was deleted, you must create a new one.

3. Distribute $INFORMIXDIR/etc/passwd_file to all the database servers that Connection Managers or the **cdr** utility connects to, and to all Connection Managers.

Note: An encrypted password file that is created on one type of operating system is not supported on a different type of operating system. On each operating system, you must run the **onpassword** utility with the same text file and encryption key.

Related tasks:

"Example: Configuring connection management for untrusted networks" on page 23-73

Modifying encrypted password information

Modify the information in the encrypted `passwd_file` file by running the **onpassword** utility.

Modify the encrypted `passwd_file` file when the following events occur:

- Database servers are added to or removed from a high-availability cluster or replication domain
- `sqlhosts` file server aliases or groups change
- User IDs or server passwords change
- You want to change your encryption key

1. Decrypt the `passwd_file` file by running **onpassword** utility, specifying the previously used encryption key and a name for the output file. For example, if you previously encrypted the file, and used **my_secret_encryption_key_asdf** as the encryption key, run the following command:

   ```
   onpassword -k my_secret_encryption_key_asdf -d my_passwords.txt
   ```

   The **onpassword** utility creates the ASCII text `my_passwords.txt` output file in the `$INFORMIXDIR/etc` directory.
2. Optional: Open the file with a text editor, and modify the information in the file.
3. Encrypt the password file with the **onpassword** utility, specifying an encryption key and the name of the text file. For example:

   ```
   onpassword -k my_secret_encryption_key_lmnop -e my_passwords.txt
   ```

   This example uses the new encryption key, **my_secret_encryption_key_lmnop**, and creates the encrypted `passwd_file` file in the `$INFORMIXDIR/etc` directory.
4. Redistribute `passwd_file` to all the database servers that the Connection Manager or **cdr** utility connects to, replacing the previous `$INFORMIXDIR/etc/passwd_file` files. If you update the `passwd_file` on multiple operating systems, you must run the **onpassword** utility on each type of operating system, and use the same text file and encryption key.

Starting Connection Managers on UNIX and Linux

Use the **oncmsm** utility to start a Connection Manager.

A Connection Manager can be started before the database servers it manages are started. If a connection unit is managed by a Connection Manager, the Connection Manager connects to database servers in the connection unit after the database servers start.

If the database servers of a connection unit are online before a Connection Manager is initialized, start the Connection Manager, and then direct client application requests to the Connection Manager instead of the database servers.

1. Start the Connection Manager by running the **oncmsm** utility with the **c** parameter and the name of the Connection Manager configuration file.

   ```
   oncmsm -c configuration_file
   ```

   The default location for the configuration file is the `$INFORMIXDIR/etc` directory.
2. If you enabled Connection Manager logging by setting the LOG and LOGFILE parameters in the Connection Manager's configuration file, check the log to verify that the Connection Manager successfully started. The following message displays is the Connection Manager started:

   ```
   Connection Manager started successfully
   ```

After a Connection Manager initializes, it attempts to connect to the database servers or groups of database servers that are specified by the INFORMIXSERVER parameter in the Connection Manager's configuration file. The Connection Manager then searches for and connects to all the database servers that are in each specified server's cluster, grid, or replicate set.

Starting Connection Managers on Windows

Use the **oncmsm** utility to start a Connection Manager.

A Connection Manager can be started before the database servers it manages are started. If a connection unit is managed by a Connection Manager, the Connection Manager connects to database servers in the connection unit after the database servers start.

If the database servers of a connection unit are online before a Connection Manager is initialized, start the Connection Manager, and then direct client application requests to the Connection Manager instead of the database servers.

1. Install the Connection Manager as a service by running the **oncmsm** utility with the **i** parameter, the **c** parameter, and the name of the Connection Manager configuration file.

   ```
   oncmsm -i -c configuration_file
   ```

   The default location for the configuration file is the `%INFORMIXDIR%\etc` directory.
2. Initialize the Connection Manager by running the **oncmsm** utility with the name of the Connection Manager that is specified in the Connection Manager configuration file.

   ```
   oncmsm connection_manager_name
   ```

   The following message is displayed:

   ```
   Specify the user and password to run this service.
   Press <ENTER> to run Connection Manager as 'localsystem'
   ```
3. Enter the **informix** user ID and password, or press **Enter** to automatically create an **informix-admin** group and assign it access rights.
4. If you enabled Connection Manager logging by setting the LOG and LOGFILE parameters in the Connection Manager's configuration file, check the log to verify that the Connection Manager successfully started. The following message displays if the Connection Manager started:

   ```
   Connection Manager started successfully
   ```

After a Connection Manager initializes, it attempts to connect to the database servers or groups of database servers that are specified by the INFORMIXSERVER parameter in the Connection Manager's configuration file. The Connection

Manager then searches for and connects to all the database servers that are in each specified server's cluster, grid, or replicate set.

Stopping connection management

When you no longer want a Connection Manager to manage connection units, run the **oncmsm** utility to stop the Connection Manager instance.

If you are using multiple Connection Managers, you can run **onstat -g cmsm** to display the names of Connection Manager instances.

1. Log on to the computer on which the Connection Manager instance is running.
2. Run the **oncmsm** utility with the **-k** parameter. For example:

   ```
   oncmsm -k connection_manager_name
   ```
3. If quality of data (QOD) monitoring is turned on, and you want to stop it, run the **cdr stop qod** command on the master server. For example:

   ```
   cdr stop qod -c server_1
   ```

The Connection Manager stops.

To unistall a stopped Connection Manager from a Windows operating system, run the following command on the computer that the Connection Manager is installed on:

```
oncmsm -u connection_manager_name
```

Monitoring and troubleshooting connection management

Tools are available to monitor connection management, and help you diagnose potential problems.

The following options are available for connection management monitoring and troubleshooting:

- Use the IBM OpenAdmin Tool (OAT) for Informix to monitor connection management.
- Set the LOG and LOGFILE parameters in Connection Manager configuration files. Log files contain information about service level agreements, failover configuration, and status information. The location of the log file is displayed when the Connection Manager is started.
- Run **onstat -g cmsm** to display information about Connection Manager instances.
- Set the CMALARMPROGRAM parameter in Connection Manager configuration files, and configure the `cmalarmprogram` script to handle event alarms.
- If the Connection Manager raises an event alarm:
  - The **INFORMIXCMNAME** environment variable stores the name of the Connection Manager instance that raised the alarm.
  - The**INFORMIXCMCONUNITNAME** environment variable stores the name of the Connection Manager connection unit that raised the alarm.

Related concepts:

Managing client connections (Informix Replication Plug-in for OAT)

Strategies for increasing availability with Connection Managers

You can increase the resiliency of your client/server communication environment.

Install multiple network-interface cards (NICs) on client, Connection Manager, and database-server hosts

You can prevent a network-connectivity failure by installing multiple NICs on each host in a connection-management domain. If a NIC fails, the host can use other available NICs.

Install multiple Connection Managers

You can prevent a Connection Manager from becoming a single point of failure in your system by installing multiple Connection Managers to manage a domain.

Install Connection Managers separate from database servers

To prevent simultaneous Connection Manager and database server failure if a host fails, install Connection Managers on hosts that are not running database servers.

Use keywords in Connection Manager service-level-agreements

If a Connection Manager manages a high-availability cluster, you can use the cluster keywords to maintain connection consistency after failover. If database servers in a cluster switch roles after failover, the Connection Manager can use cluster keywords to continue directing client connection requests to the appropriate cluster-server type.

The cluster keywords are:

- HDR - High-availability data replication server
- RSS - Remote standalone secondary server
- SDS - Shared-disk secondary server
- PRI - Primary server
- PRIMARY - Primary server

Use group entries in sqlhosts files and Connection Manager configuration files

Create database-server groups in the host `sqlhosts` file of each Connection Manager that manages a high-availability cluster. Then, specify `sqlhosts` groups, rather than individual server names, in Connection Manager configuration files. If you specify database server names for a connection unit's INFORMIXSERVER parameter, and those specified servers are offline when a Connection Manager restarts, the Connection Manager is unable to reconnect to the cluster. If you specify a `sqlhosts` group for a connection unit's INFORMIXSERVER parameter, and at least one of the group's database servers is online when a Connection Manager restarts, the Connection Manager can reconnect to the cluster.

Specify the `c=1` option for `sqlhosts` group entries so that the connection-attempt starting point for a list of group members is random. For example:

```
#dbservername   nettype    hostname   servicename   options
 my_servers     -          -                        c=1
 server_1       onsoctcp   host_1     port_1        g=my_servers
 server_2       onsoctcp   host_2     port_2        g=my_servers
 server_3       onsoctcp   host_3     port_3        g=my_servers
```

Related concepts:

"The sqlhosts information" on page 2-18

Related reference:
"Group information" on page 2-31

Configuration examples for connection management

The following examples show steps for setting up connection management for various connection units and various systems.

You can use these examples as a basis for developing your own connection-management system.

Example of configuring connection management for a high-availability cluster

This example shows steps that are required to configure connection management for a high-availability cluster.

For this example, you have a high-availability cluster on a trusted network. The cluster consists of three servers:

- A primary server (**server_1**)
- A shared-disk secondary server (**server_2**)
- An HDR secondary server (**server_3**)

The cluster supports the following application services:

- Online transaction processing (OLTP), which runs only on the primary server
- Payroll services, which can run on the primary server or HDR secondary server
- Reporting services, which can run on any of the secondary servers

Your system has the following needs:

- The database servers' workloads are balanced.
- The Connection Managers control failover.
- If failover occurs, the SD secondary server takes priority over the HDR secondary server.
- If the primary server fails, the Connection Managers can still connect to the cluster after restarting.
- The system can withstand the failure of a Connection Manager.
- The system can withstand a network-interface card (NIC) failure on each host.

To configure connection management:

1. Install at least two network interface cards on each host. This prevents the failure of a network interface card from causing Connection Manager or database server failure.
2. Install two Connection Managers. Install each Connection Manager onto a different host, and do not install the Connection Managers onto the hosts that database servers are installed on. This installation strategy prevents a Connection Manager from becoming a single point of failure, and prevents the simultaneous failure of database servers and Connection Managers if a host fails.

   You can install Connection Managers on application-server hosts if you want to prioritize an application server's connectivity to the primary cluster server.

3. On each host Connection Manager host, set the **INFORMIXDIR** environment to the directory the Connection Manager was installed into. Run the following command:

   ```
   setenv INFORMIXDIR path
   ```

4. Create a configuration file in each Connection Manager installation's $INFORMIXDIR/etc directory.

   The first Connection Manager's configuration file is named **cm_1.cfg** and has the following entries:

   ```
   NAME connection_manger_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm1_log.log
   LOCAL_IP 192.0.2.0,192.0.2.1

   CLUSTER cluster_1
   {
      INFORMIXSERVER servers_1
      SLA oltp_1    DBSERVERS=primary
      SLA payroll_1 DBSERVERS=(PRI,HDR) \
                    POLICY=WORKLOAD
      SLA report_1  DBSERVERS=(SDS,HDR) \
                    POLICY=WORKLOAD
      FOC ORDER=ENABLED \
          PRIORITY=1
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

   The second Connection Manager's configuration file is named **cm_2.cfg** and has the following entries:

   ```
   NAME connection_manger_2
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm2_log.log
   LOCAL_IP 192.0.2.2,192.0.2.3

   CLUSTER cluster_1
   {
      INFORMIXSERVER cluster_1
      SLA oltp_2    DBSERVERS=primary
      SLA payroll_2 DBSERVERS=(PRI,HDR)\
                    POLICY=WORKLOAD
      SLA report_2  DBSERVERS=(SDS,HDR) \
                    POLICY=WORKLOAD
      FOC ORDER=ENABLED \
          PRIORITY=2
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

   The configuration file specifies the following information and behavior:

   - Logging is enabled, and the log files are $INFORMIXDIR/tmp/my_cm1_log.log and $INFORMIXDIR/tmp/my_cm2_log.log.
   - **connection_manager_1** monitors 192.0.2.0 and 192.0.2.1 and **connection_manager_2** monitors 192.0.2.2 and 192.0.2.3 for network failure.
   - When the Connection Managers start, they each search their sqlhosts files for **cluster_1** entry, and then connect to the servers in that group.
   - CONNECT TO @oltp_1 and CONNECT TO @oltp_2 connection requests are directed to the primary server.
   - CONNECT TO @payroll_1 and CONNECT TO @payroll_2 connection requests are directed to whichever of the primary and HDR secondary servers has the lowest workload.
   - CONNECT TO @report_1 and CONNECT TO @report_2 connection requests are directed to the secondary server that has the lowest workload.

- The connection between **connection_manager_1** and the primary server is prioritized over the connection between **connection_manager_2** and the primary server. Failover that would break the connectivity between **connection_manager_1** and the primary server is blocked.
- If failover processing fails after eight attempts, `$INFORMIXDIR/etc/CMALARMPROGRAM.sh` is called.

Certain parameters and attributes are not included in this configuration file, so the Connection Manager has the following default behavior:

- The EVENT_TIMEOUT parameter is not set, so the Connection Managers wait 60 seconds for primary-server events before failover processing begins.
- The MODE attributes of the SLA parameters are not set, so the Connection Managers return connection information for **server_1**, **server_2**, and **server_3** to client applications, rather than acting as proxy servers.
- The SECONDARY_EVENT_TIMEOUT parameter is not set, so the Connection Managers wait 60 seconds for secondary-server events before the Connection Manager disconnects from the secondary server.
- The HOST, NETTYPE, SERVICE, and SQLHOSTSOPT attributes of the SLA parameters are not set, so each Connection Manager uses connection information in local and remote `sqlhosts` files.
- The SQLHOSTS parameter is not set, so each Connection Manager first searches its local `sqlhosts` file, and then remote database server `sqlhosts` files for connectivity information related to **server_1**, **server_2**, and **server_3**.
- The WORKERS attributes of the SLA parameters are not set, so four worker threads are allocated to each of the SLAs.

5. Set the `onconfig` file DRAUTO configuration parameter on all database servers to 3

   ```
   DRAUTO 3
   ```

   This setting specifies that a Connection Manager controls failover arbitration.
6. Set the `onconfig` file HA_FOC_ORDER configuration parameter on **server_1** to `SDS,HDR`

   ```
   HA_FOC_ORDER SDS,HDR
   ```

   After the Connection Managers start, and connect to **server_1**, the HA_FOC_ORDER value replaces the value of the `ORDER` attributes in each Connection Manager's configuration file.

   If **server_1** fails, the Connection Managers attempt failover to the SD secondary server. If the SD secondary server is also unavailable, the Connection Managers attempt failover to the HDR secondary server.
7. Optional: Configure the `cmalarmprogram` script on each Connection Manager host. Event alarms can be sent to specified email addresses.
8. Add entries to the `sqlhosts` files on **server_1** and **server_2**'s host and on **server_3**'s host.

   ```
   #dbservername  nettype    hostname   servicename   options
    server_1      onsoctcp   host_1     port_1
    server_2      onsoctcp   host_1     port_2
    server_3      onsoctcp   host_2     port_3
   ```
9. Create a `sqlhosts` file on each Connection Manager.

   ```
   #dbservername  nettype    hostname   servicename   options
    cluster_1     group      -          -             c=1,e=server_3
    server_1      onsoctcp   host_1     port_1        g=cluster_1
    server_2      onsoctcp   host_1     port_2        g=cluster_1
    server_3      onsoctcp   host_2     port_3        g=cluster_1
   ```

If a Connection Manager restarts after a primary-server failure, it is able to connect to other database servers in the cluster because the **cluster_1** group is defined.

10. Create a `sqlhosts` file on each client host.

```
#dbservername  nettype    hostname   servicename   options
 oltp          group      -          -             c=1,e=oltp_2
 oltp_1        onsoctcp   cm_host_1  cm_port_1     g=oltp
 oltp_2        onsoctcp   cm_host_2  cm_port_2     g=oltp

 report        group      -          -             c=1,e=report_2
 report_1      onsoctcp   cm_host_1  cm_port_3     g=report
 report_2      onsoctcp   cm_host_2  cm_port_4     g=report

 payroll       group      -          -             c=1,e=payroll_2
 payroll_1     onsoctcp   cm_host_1  cm_port_5     g=payroll
 payroll_2     onsoctcp   cm_host_2  cm_port_6     g=payroll
```

    If a Connection Manager fails, client applications can still connect to the other Connection Manager because the **oltp**, **report**, and **payroll** groups are defined.

    - `CONNECT TO @oltp` connection requests are directed through one of the Connection Managers to the primary server.
    - `CONNECT TO @payroll` connection requests are directed through one of the Connection Managers to whichever of the primary and HDR secondary servers has the lowest workload.
    - `CONNECT TO @report` connection requests are directed through one of the Connection Managers to the secondary server that has the lowest workload.

11. Set each **INFORMIXSQLHOSTS** environment variable to the `sqlhosts` file location by running the **setenv** command on each Connection Manager and client host.

```
setenv INFORMIXSQLHOSTS path_and_file_name
```

12. Run the **oncmsm** utility on each Connection Manager host, to start each Connection Manager.

    On the host of **connection_manager_1**:

```
oncmsm -c cm_1.cfg
```

    On the host of **connection_manager_2**:

```
oncmsm -c cm_2.cfg
```

13. Check each Connection Manager's log file to verify that the Connection Manager started correctly.

Related reference:

"Group information" on page 2-31

Example of configuring connection management for a grid or replicate set

You can use a Connection Manager to route client connections for the replication servers of a grid or replicate set.

For this example, you have a replicate set that consists of four replication servers:

- **server_1**
- **server_2**
- **server_3**
- **server_4**

The replication set supports reporting services, which can run on any of the replication servers.

Your system has the following needs

- Client requests are directed to the replication server with the fewest apply failures.
- The system can withstand the failure of a Connection Manager.
- The system can withstand a network-interface card (NIC) failure on each host.

To configure connection management:

1. Install at least two network interface cards on each host. This prevents the failure of a network interface card from causing Connection Manager or database server failure.
2. Install two Connection Managers. Install each Connection Manager onto a different host, and do not install the Connection Managers onto the hosts that database servers are installed on. This installation strategy prevents a Connection Manager from becoming a single point of failure, and prevents the simultaneous failure of database servers and Connection Managers if a host fails.
3. On each Connection Manager host, set the **INFORMIXDIR** environment to the directory the Connection Manager was installed into. Run the following command:

   ```
   setenv INFORMIXDIR path
   ```

4. Create a configuration file in each Connection Manager installation's $INFORMIXDIR/etc directory.

   The first Connection Manager's configuration file is named **cm_1.cfg** and has the following entries:

   ```
   NAME connection_manger_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm1_log.log

   REPLSET replicate_set_1
   {
    INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
    SLA report_1 DBSERVERS=ANY \
                POLICY=FAILURE
   }
   ```

   The second Connection Manager's configuration file is named **cm_2.cfg** and has the following entries:

   ```
   NAME connection_manger_2
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm2_log_.log

   REPLSET replicate_set_1
   {
    INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
    SLA report_2 DBSERVERS=ANY \
                POLICY=FAILURE
   }
   ```

   The configuration file specifies the following information and behavior:

   - Logging is enabled, and the log files are $INFORMIXDIR/tmp/my_cm1_log.log and $INFORMIXDIR/tmp/my_cm2_log.log.
   - When the Connection Managers start, they each search their sqlhosts files for **g_server_1**, **g_server_2**, **g_server_3**, and **g_server_4** entries, and then connect to the servers **server_1**, **server_2**, **server_3**, and **server_4** that are in those groups.

- `CONNECT TO @report_1` and `CONNECT TO @report_2` connection requests are directed to the replication server that has the fewest apply failures.

Certain parameters and attributes are not included in this configuration file, so the Connection Manager has the following default behavior:

- The MODE attributes of the SLA parameters are not set, so the Connection Managers return connection information for **server_1**, **server_2**, **server_3**, and **server_4** to client applications, rather than acting as proxy servers.
- The HOST, NETTYPE, SERVICE, and SQLHOSTSOPT attributes of the SLA parameters are not set, so each Connection Manager uses connection information in local and remote `sqlhosts` files.
- The SQLHOSTS parameter is not set, so each Connection Manager first searches its local `sqlhosts` file, and then remote database server `sqlhosts` files for connectivity information related to **server_1**, **server_2**, **server_3**, and **server_4**.
- The WORKERS attributes of the SLA parameters are not set, so four worker threads are allocated to each of the SLAs.

5. Add entries to the`sqlhosts` files on the hosts of each database server, **connection_manger_1**, and **connection_manger_2**.

```
#dbservername  nettype   hostname   servicename  options
 g_server_1    group     -          -            i=1,e=server_1
 server_1      onsoctcp  host_1     port_1       g=g_server_1

 g_server_2    group     -          -            i=2,e=server_2
 server_2      onsoctcp  host_2     port_2       g=g_server_2

 g_server_3    group     -          -            i=3,e=server_3
 server_3      onsoctcp  host_3     port_3       g=g_server_3

 g_server_4    group     -          -            i=4,e=server_4
 server_4      onsoctcp  host_4     port_4       g=g_server_4
```

6. Create a `sqlhosts` file on each client host.

```
#dbservername  nettype    hostname   servicename   options
 report        group      -          -             c=1,e=report_2
 report_1      onsoctcp   cm_host_1  cm_port_3     g=report
 report_2      onsoctcp   cm_host_2  cm_port_4     g=report
```

If a Connection Manager fails, client applications can still connect to the other Connection Manager because the **report** group is defined.

`CONNECT TO @report` connection requests are directed through one of the Connection Managers to the replication server that has the fewest apply failures.

7. Set each **INFORMIXSQLHOSTS** environment variable to the `sqlhosts` file location by running the **setenv** command on each Connection Manager and client host.

```
setenv INFORMIXSQLHOSTS path_and_file_name
```

8. Turn on quality of data (QOD) monitoring by running the **cdr define qod** command.

```
cdr define qod -c server_1 --start
```

The command connects to **server_1**, defines **server_1** as a master server for monitoring data, and then turns on quality of data monitoring.

server_1 maintains a failed-transaction count for the servers in the replicate set. The failed-transaction count determines which replication server the Connection Managers send a client connection requests to.

9. Run the **oncmsm** utility on each Connection Manager host, to start each Connection Manager.

On the host of **connection_manager_1**:

```
oncmsm -c cm_1.cfg
```

On the host of **connection_manager_2**:

```
oncmsm -c cm_2.cfg
```

10. Check each Connection Manager's log file to verify that the Connection Manager started correctly.

Related reference:

"Group information" on page 2-31

Example of configuring connection management for a high-availability replication system

You can use a Connection Manager to route client connections for the participants of a replicate set and to control failover for high-availability clusters that participate in Enterprise Replication.

For this example, you have a grid that consists of four nodes. One of the nodes is a primary server in a high-availability cluster that consists of a primary server, an SD secondary server, an HDR secondary server, and an RS secondary server:

- **server_1a** - ER Node 1, primary server
- **server_1b** - SD secondary server
- **server_1c** - HDR secondary server
- **server_1d** - RS secondary server
- **server_2** - ER Node 2
- **server_3** - ER Node 3
- **server_4** - ER Node 4

The grid supports reporting services, which can run on any of the ER nodes.

Your system has the following needs

- Client requests are directed to the ER node with the lowest transaction latency.
- The system can withstand the failure of a Connection Manager.
- The system can withstand a network-interface card (NIC) failure on each host.
- The Connection Managers control failover for the cluster.
- If failover occurs, it the SD secondary server takes priority over the HDR secondary server. The HDR secondary server takes priority over the RS secondary server.
- If the primary server fails, the Connection Managers can still connect to the cluster after restarting.

To configure connection management:

1. Install at least two network interface cards on each host. This prevents the failure of a network interface card from causing Connection Manager or database server failure.
2. Install two Connection Managers. Install each Connection Manager onto a different host, and do not install the Connection Managers onto the hosts that database servers are installed on. This installation strategy prevents a Connection Manager from becoming a single point of failure, and prevents the simultaneous failure of database servers and Connection Managers if a host fails.

3. On each Connection Manager host, set the **INFORMIXDIR** environment to the directory the Connection Manager was installed into. Run the following command:

   ```
   setenv INFORMIXDIR path
   ```

4. Create a configuration file in each Connection Manager installation's $INFORMIXDIR/etc directory.

   The first Connection Manager's configuration file is named **cm_1.cfg** and has the following entries:

   ```
   NAME connection_manger_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm1_log.log
   LOCAL_IP 192.0.0.2,192.0.2.1

   REPLSET replicate_set_1
   {
    INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
    SLA report_1 DBSERVERS=ANY \
                POLICY=LATENCY
   }

   CLUSTER cluster_1
   {
      INFORMIXSERVER g_server_1
      FOC ORDER=ENABLED \
          PRIORITY=1
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

   The second Connection Manager's configuration file is named **cm_2.cfg** and has the following entries:

   ```
   NAME connection_manger_2
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm2_log_.log
   LOCAL_IP 192.0.2.2,192.0.2.3

   REPLSET replicate_set_1
   {
    INFORMIXSERVER g_server_1,g_server_2,g_server_3,g_server_4
    SLA report_2 DBSERVERS=ANY \
                POLICY=LATENCY
   }
   CLUSTER cluster_1
   {
      INFORMIXSERVER g_server_1
      FOC ORDER=ENABLED \
          PRIORITY=2
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

   The configuration file specifies the following information and behavior:

   - Logging is enabled, and the log files are $INFORMIXDIR/tmp/my_cm1_log.log and $INFORMIXDIR/tmp/my_cm2_log.log.
   - **connection_manager_1** monitors 192.0.2.0 and 192.0.2.1 and **connection_manager_2** monitors 192.0.2.2 and 192.0.2.3 for network failure.
   - When the Connection Managers start, they each search their sqlhosts files for **g_server_1**, **g_server_2**, **g_server_3**, and **g_server_4** entries, and then connect to the servers **server_1a**, **server_1b**, **server_1c**, **server_1d**,**server_2**, **server_3**, and **server_4** that are in those groups.
   - CONNECT TO @report_1 and CONNECT TO @report_2 connection requests are directed to the replication server that has the lowest transaction latency.

- The connection between **connection_manager_1** and the primary server is prioritized over the connection between **connection_manager_2** and the primary server. Failover that would break the connectivity between **connection_manager_1** and the primary server is blocked.
- If failover processing fails after eight attempts, `$INFORMIXDIR/etc/CMALARMPROGRAM.sh` is called.

Certain parameters and attributes are not included in this configuration file, so the Connection Manager has the following default behavior:

- The EVENT_TIMEOUT parameter is not set, so the Connection Managers wait 60 seconds for primary-server events before failover processing begins. The SECONDARY_EVENT_TIMEOUT parameter is not set, so the Connection Managers wait 60 seconds for secondary-server events before the Connection Manager disconnects from the secondary server.
- The HOST, NETTYPE, SERVICE, and SQLHOSTSOPT attributes of the SLA parameters are not set, so each Connection Manager uses connection information in local and remote `sqlhosts` files.
- The SQLHOSTS parameter is not set, so each Connection Manager first searches its local `sqlhosts` file, and then remote database server `sqlhosts` files for connectivity information related to **server_1**, **server_2**, **server_3**, and **server_4**.
- The WORKERS attributes of the SLA parameters are not set, so four worker threads are allocated to each of the SLAs.

5. Set the `onconfig` file DRAUTO configuration parameter on **server_1a**, **server_1b**, **server_1c**, and **server_1d** to 3

   ```
   DRAUTO 3
   ```

   This setting specifies that a Connection Manager controls failover arbitration.
6. Set the `onconfig` file HA_FOC_ORDER configuration parameter on **server_1a** to `SDS,HDR,RSS`

   ```
   HA_FOC_ORDER SDS,HDR,RSS
   ```

   After the Connection Managers start, and connect to **server_1a**, the HA_FOC_ORDER value replaces the value of the `ORDER` attributes in each Connection Manager's configuration file.

   If **server_1a** fails, the Connection Managers attempt failover to the SD secondary server. If the SD secondary server is also unavailable, the Connection Managers attempt failover to the HDR secondary server. If the HDR secondary server is also unavailable, the Connection Managers attempt failover to the RS secondary server.
7. Add entries to the`sqlhosts` files on the hosts of each database server and Connection Manager.

   ```
   #dbservername  nettype   hostname  servicename  options
    g_server_1    group     -         -            i=1,c=1,e=server_1d
    server_1a     onsoctcp  host_1    port_1       g=g_server_1
    server_1b     onsoctcp  host_1    port_2       g=g_server_1
    server_1c     onsoctcp  host_2    port_3       g=g_server_1
    server_1d     onsoctcp  host_3    port_4       g=g_server_1

    g_server_2    group     -         -            i=2,e=server_2
    server_2      onsoctcp  host_4    port_5       g=g_server_2

   g_server_3    group     -         -            i=3,e=server_3
    server_3      onsoctcp  host_5    port_6       g=g_server_3

    g_server_4    group     -         -            i=4,e=server_4
    server_4      onsoctcp  host_6    port_7       g=g_server_4
   ```

8. Create a sqlhosts file on each client host.

```
#dbservername  nettype    hostname   servicename   options
 report        group      -          -             c=1,e=report_2
 report_1      onsoctcp   cm_host_1  cm_port_3     g=report
 report_2      onsoctcp   cm_host_2  cm_port_4     g=report
```

   If a Connection Manager fails, client applications can still connect to the other Connection Manager because the **report** group is defined.

   CONNECT TO @report connection requests are directed through one of the Connection Managers to the replication server that has the lowest transaction latency.

9. Set each **INFORMIXSQLHOSTS** environment variable to the sqlhosts file location by running the **setenv** command on each Connection Manager and client host.

   ```
   setenv INFORMIXSQLHOSTS path_and_file_name
   ```

10. Turn on quality of data (QOD) monitoring by running the **cdr define qod** command.

    ```
    cdr define qod -c server_1a --start
    ```

    The command connects to **server_1a**, defines **server_1a** as a master server for monitoring data, and then turns on quality of data monitoring.

    server_1a monitors transaction latency for the replication servers in the grid.

11. Run the **oncmsm** utility on each Connection Manager host, to start each Connection Manager.

    On the host of **connection_manager_1**:

    ```
    oncmsm -c cm_1.cfg
    ```

    On the host of **connection_manager_2**:

    ```
    oncmsm -c cm_2.cfg
    ```

12. Check each Connection Manager's log file to verify that the Connection Manager started correctly.

Related reference:

"Group information" on page 2-31

Example: Configuring connection management for untrusted networks

This example shows steps that are required to configure connection management for an untrusted network.

For this example, you have a high-availability cluster on an untrusted network. All hosts use UNIX operating systems. The cluster consists of four servers:

- A primary server (**server_1**)
- A shared-disk secondary server (**server_2**)
- An HDR secondary server (**server_3**)
- An RS secondary server (**server_4**)

To configure connection management:

1. Install at least two network interface cards on each host.
2. Install at least two Connection Managers. Install each Connection Manager onto a different host, and do not install the Connection Managers onto the hosts that database servers are installed on.

3. On each host Connection Manager host, set the **INFORMIXDIR** environment to the directory the Connection Manager was installed into. Run the following command:

   ```
   setenv INFORMIXDIR path
   ```

4. Create a configuration file in each Connection Manager installation's $INFORMIXDIR/etc directory.

   The first Connection Manager's configuration file is named **cm_1.cfg** and has the following entries:

   ```
   NAME connection_manger_1
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm1_log.log
   LOCAL_IP 192.0.2.0,192.0.2.1

   CLUSTER cluster_1
   {
      INFORMIXSERVER cluster_1
      SLA oltp_1    DBSERVERS=primary
      SLA payroll_1 DBSERVERS=(PRI,HDR) \
                    POLICY=WORKLOAD
      SLA report_1  DBSERVERS=(SDS,HDR,RSS) \
                    POLICY=WORKLOAD
      FOC ORDER=ENABLED \
          PRIORITY=1
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

   The second Connection Manager's configuration file is named **cm_2.cfg** and has the following entries:

   ```
   NAME connection_manger_2
   LOG 1
   LOGFILE $INFORMIXDIR/tmp/my_cm2_log.log
   LOCAL_IP 192.0.2.2,192.0.2.3

   CLUSTER cluster_1
   {
      INFORMIXSERVER cluster_1
      SLA oltp_2    DBSERVERS=primary
      SLA payroll_2 DBSERVERS=(PRI,HDR)\
                    POLICY=WORKLOAD
      SLA report_2  DBSERVERS=(SDS,HDR,RSS) \
                    POLICY=WORKLOAD
      FOC ORDER=ENABLED \
          PRIORITY=2
      CMALARMPROGRAM $INFORMIXDIR/etc/CMALARMPROGRAM.sh
   }
   ```

5. Set the onconfig file DRAUTO configuration parameter on all database servers to 3, to specify that Connection Managers control failover arbitration.

   ```
   DRAUTO 3
   ```

6. Set the onconfig file HA_FOC_ORDER configuration parameter on **server_1** to SDS,HDR,RSS

   ```
   HA_FOC_ORDER SDS,HDR,RSS
   ```

7. Optional: Configure the cmalarmprogram script on each Connection Manager host.

8. Add entries to thesqlhosts files on **server_1** and **server_2**'s host, **server_3**'s host, and **server_4**'s host.

   ```
   #dbservername   nettype    hostname   servicename   options
    server_1       onsoctcp   host_1     port_1        s=6
    a_server_1     onsoctcp   host_1     port_2

    server_2       onsoctcp   host_1     port_3        s=6
   ```

```
a_server_2     onsoctcp   host_1     port_4

server_3       onsoctcp   host_2     port_5        s=6
a_server_3     onsoctcp   host_2     port_6

server_4       onsoctcp   host_3     port_7        s=6
a_server_4     onsoctcp   host_3     port_8
```

9. Create a sqlhosts file on each Connection Manager's host.

```
#dbservername   nettype    hostname   servicename   options
 cluster_1      group      -          -             c=1,e=a_server_4
 server_1       onsoctcp   host_1     port_1        s=6,g=cluster_1
 a_server_1     onsoctcp   host_1     port_2        g=cluster_1
 server_2       onsoctcp   host_1     port_3        s=6,g=cluster_1
 a_server_2     onsoctcp   host_1     port_4        g=cluster_1
 server_3       onsoctcp   host_2     port_5        s=6,g=cluster_1
 a_server_3     onsoctcp   host_2     port_6        g=cluster_1
 server_4       onsoctcp   host_3     port_7        s=6,g=cluster_1
 a_server_4     onsoctcp   host_3     port_8        g=cluster_1
```

10. In each database server's onconfig file, set the DBSERVERALIASES parameter to that database server's alias.

    The onconfig file entry for **server_1**:

    ```
    DBSERVERALIASES a_server_1
    ```

    The onconfig file entry for **server_2**:

    ```
    DBSERVERALIASES a_server_2
    ```

    The onconfig file entry for **server_3**:

    ```
    DBSERVERALIASES a_server_3
    ```

    The onconfig file entry for **server_4**:

    ```
    DBSERVERALIASES a_server_4
    ```

11. On one of the Connection Manager hosts, use a text editor to create an ASCII-text password file that contains security information. Save the file to the $INFORMIXDIR/tmp directory. For example, my_passwords.txt has the following entries:

```
cluster_1   a_server_1  user_1  password_1
cluster_1   a_server_2  user_2  password_2
cluster_1   a_server_3  user_3  password_3
cluster_1   a_server_4  user_4  password_4

server_1    a_server_1  user_1  password_1
server_2    a_server_2  user_2  password_2
server_3    a_server_3  user_3  password_3
server_4    a_server_4  user_4  password_4

a_server_1  a_server_1  user_1  password_1
a_server_2  a_server_2  user_2  password_2
a_server_3  a_server_3  user_3  password_3
a_server_4  a_server_4  user_4  password_4
```

12. On the host where the password file is saved, run the **onpassword** utility with a specified encryption key to encrypt the password and create passwd_file in the $INFORMIXDIR/etc directory. For example, run the following command, specifying **my_secret_encryption_key_456** as your encryption key:

    ```
    onpassword -k my_secret_encryption_key_456 -e my_passwords.txt
    ```

13. Store the original text file and encryption key in a safe place.
14. Distribute $INFORMIXDIR/etc/passwd_file to all the database servers that Connection Managers connect to, and to all Connection Managers. For systems that use Enterprise Replication, also distribute $INFORMIXDIR/etc/passwd_file to all the database servers that the **cdr** utility connects to.
15. Create a sqlhosts file on each client host.

```
#dbservername  nettype    hostname   servicename   options
 oltp          group      -          -             c=1,e=oltp_2
 oltp_1        onsoctcp   cm_host_1  cm_port_1     g=oltp
 oltp_2        onsoctcp   cm_host_2  cm_port_2     g=oltp

 report        group      -          -             c=1,e=report_2
 report_1      onsoctcp   cm_host_1  cm_port_3     g=report
 report_2      onsoctcp   cm_host_2  cm_port_4     g=report

 payroll       group      -          -             c=1,e=payroll_2
 payroll_1     onsoctcp   cm_host_1  cm_port_5     g=payroll
 payroll_2     onsoctcp   cm_host_2  cm_port_6     g=payroll
```

16. Set each **INFORMIXSQLHOSTS** environment variable to the `sqlhosts` file location by running the **setenv** command on each Connection Manager and client host.

    ```
    setenv INFORMIXSQLHOSTS path_and_file_name
    ```

17. Run the **oncmsm** utility on each Connection Manager host, to start each Connection Manager.

    On the host of **connection_manager_1**:

    ```
    oncmsm -c cm_1.cfg
    ```

    On the host of **connection_manager_2**:

    ```
    oncmsm -c cm_2.cfg
    ```

18. Check each Connection Manager's log file to verify that the Connection Manager started correctly.

Related tasks:

"Creating a password file for connecting to database servers on untrusted networks" on page 23-58

Related reference:

"Group information" on page 2-31

Example of configuring connection management for prioritizing connections and network monitoring

You can install Connection Managers on the hosts of application servers, and then prioritize the connections between specific application servers and the primary server of a high-availability cluster. This configuration allows the highest priority application server to maintain its connection to the cluster's primary server if a portion of the network fails.

You can configure failover for the following conditions:

- When the primary server becomes inoperative.
- When an application server loses network connectivity with the primary server.

If network monitoring and failover priority are enabled and a network failure occurs, an application server that loses connectivity to the primary server but maintains connectivity to a secondary server can, through a shared-host Connection Manager, initiate failover to the secondary server. If, however, failover would cause an application server with more priority to lose connectivity to the primary server, the Connection Managers block failover.

Network monitoring and failover priority are enabled by setting the following parameters and attributes in each Connection Manager's configuration file:

- The Connection Manager's LOCAL_IP parameter
- A CLUSTER connection-unit's SLA parameters
- A CLUSTER connection-unit's FOC parameter

- The FOC parameter's ORDER attribute
- The FOC parameter's PRIORITY attribute

1. Install at least two network interface cards on each host. This method prevents the failure of a network interface card from causing client, Connection Manager, or database server connectivity failure.
2. Install and configure Connection Managers on each application server's host.
   a. Set the LOCAL_IP parameter in each Connection Manager configuration file to the IP addresses of the host's NIC cards. For example:

      The first Connection Manager's configuration file is named **cm_1.cfg** and has the following entries:

      ```
      NAME connection_manager_1
      LOCAL_IP 192.0.2.0,192.0.2.1
      ```

      The second Connection Manager's configuration file is named **cm_2.cfg** and has the following entries:

      ```
      NAME connection_manager_2
      LOCAL_IP 192.0.2.2,192.0.2.3
      ```

   b. Create service-level agreements for each Connection Manager. For example:

      cm_1.cfg now has the following entries:

      ```
      NAME connection_manager_1
      LOCAL_IP 192.0.2.0,192.0.2.1

      CLUSTER my_cluster

      {
         INFORMIXSERVER my_servers
         SLA sla_1 DBSERVERS=PRI
         SLA sla_2 DBSERVERS=SDS,HDR,RSS
      }
      ```

      cm_2.cfg now has the following entries:

      ```
      NAME connection_manager_1
      LOCAL_IP 192.0.2.2,192.0.2.3

      CLUSTER my_cluster

      {
         INFORMIXSERVER my_servers
         SLA sla_3 DBSERVERS=PRI
         SLA sla_4 DBSERVERS=SDS,HDR,RSS
      }
      ```

   c. Specify each application server's priority by setting the FOC parameter's PRIORITY attribute for each shared-host Connection Manager. A PRIORITY value must be a positive integer and unique among all the Connection Managers that are configured to manage a specific cluster. If you specify a PRIORITY value in a connection-unit definition, you must set the ORDER attribute to ENABLED, and specify the failover order in the primary server's HA_FOC_ORDER configuration parameter. For example:

      The primary server has the following `onconfig` file entry:

      ```
      HA_FOC_ORDER SDS,HDR,RSS
      ```

      cm_1.cfg now has the following entries:

      ```
      NAME connection_manager_1
      LOCAL_IP 192.0.2.0,192.0.2.1

      CLUSTER my_cluster

      {
         INFORMIXSERVER my_servers
      ```

```
   SLA sla_1 DBSERVERS=PRI
   SLA sla_2 DBSERVERS=SDS,HDR,RSS
   FOC ORDER=ENABLED PRIORITY=1
}
```

cm_2.cfg now has the following entries:

```
NAME connection_manager_2
LOCAL_IP 192.0.2.2,192.0.2.3

CLUSTER my_cluster

{
   INFORMIXSERVER my_servers
   SLA sla_3 DBSERVERS=PRI
   SLA sla_4 DBSERVERS=SDS,HDR,RSS
   FOC ORDER=ENABLED PRIORITY=2
}
```

3. Create `sqlhosts` files that contain network connectivity information for each Connection Manager and database server host.
4. Optional: Create a password file if you configure secure ports for database servers.
5. Run the **oncmsm** utility on each Connection Manager host, to start each Connection Manager.

   On the host of **connection_manager_1**:

   ```
   oncmsm -c cm_1.cfg
   ```

   On the host of **connection_manager_2**:

   ```
   oncmsm -c cm_2.cfg
   ```
6. Check each Connection Manager's log file to verify that the Connection Manager started correctly.

The Connection Managers now initiate failover if a network failure occurs, and failover would not cause a higher-priority application server to lose its connectivity to the cluster's primary server.

Related reference:

"Group information" on page 2-31

"FOC Connection Manager configuration parameter" on page 23-11

"LOCAL_IP Connection Manager configuration parameter" on page 23-17