Managing High Availability Installations

This section describes how to set up IoT FND for high availability, and includes the following sections:

Overview of IoT FND High Availability

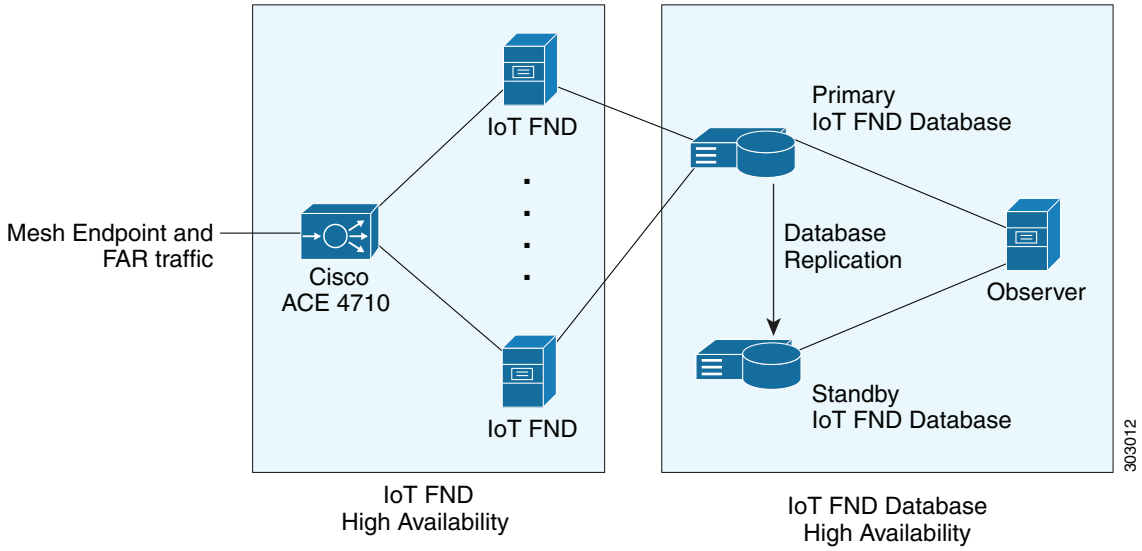
This section provides an overview of IoT FND high availability installations, including the following sections:

IoT FND is a critical application for monitoring and managing a connected grid. IoT FND High Availability (IoT FND HA) solutions address the overall availability of IoT FND during software, network, or hardware failures.

IoT FND provides two main levels of HA, as shown in Figure 1:

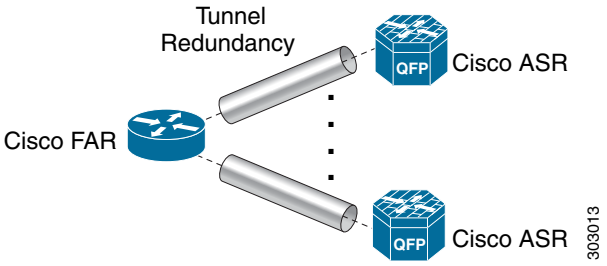
- IoT FND Server HA—This is achieved by connecting multiple IoT FND servers to a Cisco ACE 4710 load balancer. Traffic originating at MEs, FARs, and ASRs goes to the load balancer, which uses a round-robin protocol to distribute the load among the IoT FND cluster servers.
- IoT FND Database HA—This is achieved by configuring two IoT FND Database servers: a primary server and a standby (or secondary) server. When the primary database receives new data it sends a copy to the standby database. A separate system runs the Observer (the Observer can also run on the standby server), which is a program that monitors the IoT FND Database servers. If the primary database fails, the Observer configures the standby server as the new primary database. IoT FND Database HA works in single and cluster IoT FND server deployments.

Figure 1 IoT FND Server and Database HA



In addition to IoT FND Server and Database HA, IoT FND improves reliability by adding tunnel redundancy. This is achieved by defining multiple tunnels between one FAR and multiple ASRs. If one tunnel fails, the FAR routes traffic through another tunnel.

Figure 2 IoT FND Tunnel Redundancy



IoT FND HA addresses these failure scenarios:

Failure Type	Description
IoT FND server failure	If a server within a IoT FND server cluster fails, the load balancer routes traffic to the other servers in the cluster.
IoT FND database failures	If the primary database fails, the associated standby database becomes the primary database. This is transparent to the IoT FND servers. All IoT FND servers in the cluster connect to the new primary database.
Tunnel failure	If a tunnel fails, traffic flows through another tunnel.

Load Balancer

The Load Balancer (LB) plays a critical role in IoT FND HA, as it performs these tasks:

- Load balances traffic destined for IoT FND.
- Maintains heartbeats with servers in the cluster and detects any failure. If a IoT FND server fails, the LB directs traffic to other cluster members.

Cisco recommends using the Cisco ACE 4710 (Cisco ACE) as the load balancer in this deployment. See http://www.cisco.com/en/US/partner/products/ps7027/tsd_products_support_series_home.html for information on the Cisco ACE 4710.

Server Heartbeats

The LB maintains heartbeats with each IoT FND server in the cluster. In the health monitoring mechanism adopted by the IoT FND solution (there are alternate solutions), the heartbeats are regular GET messages to IoT FND on port 80. IoT FND expects an HTTP 200 OK response from an active IoT FND server.

You can configure these heartbeat parameters on the LB:

- Periodicity of probes—This is the number of seconds between heartbeats. The default value on the Cisco ACE is 15 seconds
- Number of retries—This is the number of times the LB tries to send a heartbeat to a non-responding IoT FND server before declaring it down. The default number of retries is 3.
- Regular checks after failure detection—The LB checks whether the server is back online at this time interval. The default failure detection check value is 60 seconds.

Database High Availability

IoT FND Database HA works in IoT FND single-server and cluster deployments. IoT FND HA uses Oracle Active Dataguard to deploy Oracle HA. To configure HA for the IoT FND Database, use the Oracle Recovery Manager (RMAN) and Dataguard Management CLI (DGMGRL).

The IoT FND Database HA configuration process involves:

- Configuring the primary and secondary databases the same on separate physical servers.

  Note: The secondary database server is also referred to as the standby database.

  Note: There is a possibility of losing some data during a database failover.

- Configuring data replication to be performed over SSL using an Oracle *wallet*. The wallet contains a self-signed certificate to facilitate quick deployment.

  Note: The Oracle wallet bundled with the IoT FND RPMs uses self-signed certificates. You can configure custom certificates and wallet to facilitate replication.

  Note: There is no performance impact when performing data replication over SSL.

- Using the sys user for replication and *not* cgms_dev.
- Configuring replication as asynchronous to prevent performance bottlenecks.

By default, IoT FND connects to the database using TCP over port 1522. Replication uses TCPS (TCP over SSL) on port 1622.

The scripts for configuring IoT FND Database HA are included in the IoT FND Oracle Database RPM package (cgms-oracle-*version_number*.x86_64.rpm). When you install the IoT FND Database, the HA scripts are located in $ORACLE_HOME/cgms/scripts/ha.

Tunnel Redundancy

To add another layer of redundancy to your IoT FND deployment, configure multiple tunnels to connect every FAR in a FAR tunnel provisioning group to multiple ASRs. For example, you could configure IoT FND to provision two tunnels for every FAR. One tunnel is active over the Cellular interface, while the redundant tunnel is configured to communicate with a second ASR over the WiMAX interfaces.

To configure tunnel redundancy, you need to:

1. Add ASRs to a tunnel provisioning group.
2. Modify the tunnel provisioning templates to include commands to create additional tunnels.
3. Define policies that determine the mapping between interfaces on the FAR and ASR interfaces:
   - Configuring Tunnel Provisioning Policies
   - Modifying the Tunnel Provisioning Templates for Tunnel Redundancy

HA Guidelines and Limitations

Note the following about IoT FND HA configurations:

- IoT FND HA does not include HA support for other network components like FARs, ASRs, and the load balancer.
- Zero service downtime is targeted by IoT FND HA, but it is not guaranteed.
- All IoT FND nodes must on the same subnet.
- All IoT FND nodes must run on similar hardware.
- All IoT FND nodes must run the same software version.
- Run the IoT FND setup script (/opt/cgms/bin/setupCgms.sh) on all the nodes.
- Run the DB migration script (/opt/cgms/bin/db-migrate) on only one node.
- The /opt/cgms/bin/print_cluster_view.sh script displays information about IoT FND cluster members.

HA Installation for FND

Be sure to enter the following information in the /opt/cgms/bin/cgms.conf file:

CLUSTER_BIND_ADDR= a.b.c.d

UDP_MULTICAST_ADDR= w.x.y.z

where CLUSTER_BIND_ADDR is the IP address of the server itself and UDP_MULTICAST_ADDR must be the same on all instances. It can be either an IPv4 multicast address or an IPv6 address which is not used in the network.

Configuring IoT FND Installations for HA

This section describes the various configuration settings for IoT FND HA installations, including the following sections:

- Setting Up IoT FND Database for HA
- Disabling IoT FND Database HA
- Load-Balancing Policies

- Running LB Configuration Example
- Configuring Tunnel Provisioning Policies
- Modifying the Tunnel Provisioning Templates for Tunnel Redundancy

Setting Up IoT FND Database for HA

To set up the IoT FND Database HA:

1. Set up the standby database (see Setting Up the Standby Database).

   Note: Always configure the standby database first.

   - The default SID for the standby server is **cgms_s** and *not* cgms.
   - Before setting up the standby server for HA, ensure that the environment variable $ORACLE_SID on the standby server is set to **cgms_s**.
   - The port is always 1522.

2. Set up the primary database (see Setting Up the Primary Database).

   - The default SID for the primary server is **cgms**.
   - Before setting up the primary server for HA, ensure that the environment variable $ORACLE_SID on the primary server is set to **cgms**.

3. Set up IoT FND for database HA (see Setting Up IoT FND for Database HA).
4. Set up the database Observer (see Setting Up the Observer).

Setting Up the Standby Database

To set up the standby database server for HA, run the setupStandbyDb.sh script. This script prompts for configuration information needed for the standby database, including the IP address of the primary database.

```
$ ./setupStandbyDb.sh
$ Are you sure you want to setup a standby database ? (y/n)? y

09-20-2012 13:59:18 PDT: INFO: User response: y
09-20-2012 13:59:18 PDT: INFO: CGMS_S database does not exist.
Enter the SYS DBA password. NOTE: This password should be same as the one set on the primary server:
Re-enter password for SYS DBA:
09-20-2012 13:59:58 PDT: INFO: User entered SYS DBA password.

Enter new password for CG-NMS database:
Re-enter new password CG-NMS database:
09-20-2012 14:00:09 PDT: INFO: User entered CG-NMS DB password.
Enter primary database server IP address: 192.168.1.12
09-20-2012 14:00:27 PDT: INFO: Cleaning up instance - cgms_s
DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production
…
Total System Global Area  329895936 bytes
Fixed Size    2228024 bytes
Variable Size  255852744 bytes
Database Buffers   67108864 bytes
Redo Buffers    4706304 bytes
...
```

```
09-20-2012 14:00:29 PDT: INFO: ========== CGMS_S Database Setup Completed Successfully ==========
```

Setting Up the Primary Database

To set up the primary database server for HA, run the setupHaForPrimary.sh script. This script prompts for configuration information needed for the primary database, including the IP address of the standby database.

```
$ ./setupHaForPrimary.sh
[oracle@pdb ha]$ ./setupHaForPrimary.sh
09-20-2012 13:58:39 PDT: INFO: ORACLE_BASE: /home/oracle/app/oracle
09-20-2012 13:58:39 PDT: INFO: ORACLE_HOME: /home/oracle/app/oracle/product/11.2.0/dbhome_1
09-20-2012 13:58:39 PDT: INFO: ORACLE_SID : cgms
09-20-2012 13:58:39 PDT: INFO: Make sure the above environment variables are what you expect

Are you sure you wish to configure high availability for this database server ? (y/n)? y

09-20-2012 13:58:45 PDT: INFO: User response: y
Enter standby database server IP address: 192.168.1.10
09-20-2012 13:58:56 PDT: INFO: Secondary listener reachable. Moving on with configuration
mkdir: cannot create directory `/home/oracle/app/oracle/oradata/cgms': File exists
09-20-2012 13:58:58 PDT: INFO: Reloading the listener to pick the new settings

LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 20-SEP-2012 13:58:58

…
DGMGRL> 09-20-2012 14:14:54 PDT: INFO: Please start the 'Observer' on appropriate server for ha
monitoring
Total time taken to perform the operation: 975 seconds
09-20-2012 14:14:54 PDT: INFO: ==========  Completed Successfully ==========
```

Setting Up the Observer

The Observer should run on a separate server, but can be set up on the server hosting the standby database.

Note: The password required for running Observer is the same as the SYS DBA password. See Creating the IoT FND Oracle Database

To set up the Observer:

1. On a separate server, run the observer script.

   ```
   $ ./manageObserver.sh start cgms_s password
   $ DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production
   ...
   Welcome to DGMGRL, type "help" for information.
   DGMGRL> Connected.
   DGMGRL> Observer started
   ```

2. Run the getHaStatus.sh script to verify that the database is set up for HA.

   ```
   $ ./getHaStatus.sh
   ...
   Configuration - cgms_dgconfig

     Protection Mode: MaxPerformance
     Databases:
       cgms   - Primary database
       cgms_s - (*) Physical standby database

   Fast-Start Failover: ENABLED

   Configuration Status:
   SUCCESS
   ```

```
DGMGRL>
Database - cgms

  Role:          PRIMARY
  Intended State:  TRANSPORT-ON
  Instance(s):
    cgms

Database Status:
SUCCESS

DGMGRL>
Database - cgms_s

  Role:          PHYSICAL STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds
  Apply Lag:      0 seconds
  Real Time Query: OFF
  Instance(s):
    cgms_s

Database Status:
SUCCESS
```

Setting Up IoT FND for Database HA

To set up IoT FND for database HA:

1. Stop IoT FND.
2. Run the setupCgms.sh script.

   The script prompts you to change the database settings. Enter **y**. Then, the script prompts you to enter the primary database server information (IP address, port, and database SID). After that, the script prompts you to add another database server. Enter **y**. Then, the script prompts you to enter the standby database server information (IP address, port, and database SID), as follows:

   Note: IoT FND always uses port 1522 to communicate with the database. Port 1622 is only used by the database for replication.

```
# cd /opt/cgms/bin
# ./setupCgms.sh
09-13-2012 17:10:00 PDT: INFO: ========== CG-NMS Setup Started - 2012-09-13-17-10-00 ==========
09-13-2012 17:10:00 PDT: INFO: Log file: /opt/cgms/bin/../server/cgms/log/cgms_setup.log

Are you sure you want to setup CG-NMS (y/n)? y

09-13-2012 17:10:02 PDT: INFO: User response: y

Do you want to change the database settings (y/n)? y

09-13-2012 17:10:05 PDT: INFO: User response: y

Enter database server IP address [128.107.154.246]: 128.107.154.246
09-13-2012 17:11:02 PDT: INFO: Database server IP: 128.107.154.246

Enter database server port [1522]:
09-13-2012 17:11:07 PDT: INFO: Database server port: 1522
```

```
Enter database SID [cgms]:
09-13-2012 17:11:12 PDT: INFO: Database SID: cgms

Do you wish to configure another database server for this CG-NMS ? (y/n)? y

09-13-2012 17:11:18 PDT: INFO: User response: y
Enter database server IP address []: 128.107.154.20
09-13-2012 17:11:02 PDT: INFO: Database server IP: 128.107.154.20
Enter database server port []: 1522
09-13-2012 17:11:07 PDT: INFO: Database server port: 1522
Enter database SID []: cgms_s
09-13-2012 17:11:12 PDT: INFO: Database SID: cgms_s
09-13-2012 17:11:18 PDT: INFO: Configuring database settings. This may take a while. Please wait
...
09-13-2012 17:11:19 PDT: INFO: Database settings configured.

Do you want to change the database password (y/n)? y

09-13-2012 17:15:07 PDT: INFO: User response: y

Enter database password:
Re-enter database password:

09-13-2012 17:15:31 PDT: INFO: Configuring database password. This may take a while. Please wait
...
09-13-2012 17:15:34 PDT: INFO: Database password configured.

Do you want to change the keystore password (y/n)? n

09-13-2012 17:16:18 PDT: INFO: User response: n

Do you want to change the web application 'root' user password (y/n)? n

09-13-2012 17:16:34 PDT: INFO: User response: n

Do you want to change the FTP settings (y/n)? n

09-13-2012 17:16:45 PDT: INFO: User response: n
09-13-2012 17:16:45 PDT: INFO: ========== CG-NMS Setup Completed Successfully ==========
```

Disabling IoT FND Database HA

To disable IoT FND Database HA:

1. On the server running the Observer program, stop the Observer:

```
$ ./manageObserver.sh stop cgms_s password
DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> Connected.
DGMGRL> Done.
$ Observer stopped
```

2. On the standby IoT FND Database server, delete the standby database:

```
$ ./deleteStandbyDb.sh

Are you sure you want to delete the standby database ? All replicated data will be lost (y/n)? y

09-20-2012 14:27:02 PDT: INFO: User response: y
09-20-2012 14:27:02 PDT: INFO: Cleaning up instance - cgms_s
```

```
DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> Connected.
DGMGRL> Done.
DGMGRL> DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> Connected.
DGMGRL> Disabled.
DGMGRL> 09-20-2012 14:27:06 PDT: INFO: Removing dataguard configuration
DGMGRL for Linux: Version 11.2.0.3.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
DGMGRL> Connected.
DGMGRL> Removed configuration
DGMGRL> 09-20-2012 14:27:07 PDT: INFO: Stopping the database

SQL*Plus: Release 11.2.0.3.0 Production on Thu Sep 20 14:27:07 2012

Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> ORA-01109: database not open


Database dismounted.
ORACLE instance shut down.
SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 20-SEP-2012 14:27:19

Copyright (c) 1991, 2011, Oracle.  All rights reserved.

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=test-scale-15krpm)(PORT=1522))(CONNECT_DATA=(SERVER=DEDIC
ATED)(SERVICE_NAME=cgms_s)))
The command completed successfully
Cleaning up instance - cgms_s
09-20-2012 14:27:29 PDT: INFO: ==========  Completed Successfully ==========
```

3. On the primary IoT FND Database server, delete the HA configuration:

```
$ ./deletePrimaryDbHa.sh
Are you sure you want to delete the high availability configuration ? All replicated data will be
lost (y/n)? y

09-20-2012 14:25:25 PDT: INFO: User response: y
09-20-2012 14:25:25 PDT: INFO: Removing secondary configuration from primary

SQL*Plus: Release 11.2.0.3.0 Production on Thu Sep 20 14:25:25 2012
```

```
Copyright (c) 1982, 2011, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>
System altered.
...
SQL> Disconnected from Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
09-20-2012 14:25:28 PDT: INFO: Removing data guard config files
09-20-2012 14:25:28 PDT: INFO: Removing standby redo logs
09-20-2012 14:25:29 PDT: INFO: Creating listener file
09-20-2012 14:25:29 PDT: INFO: Listener successfully configured.
09-20-2012 14:25:29 PDT: INFO: Recreating tnsnames ora file
09-20-2012 14:25:29 PDT: INFO: reloading the listener

LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 20-SEP-2012 14:25:29

Copyright (c) 1991, 2011, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=test-scale-15krpm-db2)(PORT=1522)))
The command completed successfully

LSNRCTL for Linux: Version 11.2.0.3.0 - Production on 20-SEP-2012 14:25:30

Copyright (c) 1991, 2011, Oracle.  All rights reserved.

Starting /home/oracle/app/oracle/product/11.2.0/dbhome_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 11.2.0.3.0 - Production
System parameter file is /home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora
Log messages written to
/home/oracle/app/oracle/diag/tnslsnr/test-scale-15krpm-db2/cgmstns/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=test-scale-15krpm-db2)(PORT=1522)))

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=test-scale-15krpm-db2)(PORT=1522)))
STATUS of the LISTENER
------------------------
Alias                     cgmstns
Version                   TNSLSNR for Linux: Version 11.2.0.3.0 - Production
Start Date                20-SEP-2012 14:25:30
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File
/home/oracle/app/oracle/product/11.2.0/dbhome_1/network/admin/listener.ora
Listener Log File
/home/oracle/app/oracle/diag/tnslsnr/test-scale-15krpm-db2/cgmstns/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=test-scale-15krpm-db2)(PORT=1522)))
Services Summary...
Service "cgms" has 1 instance(s).
  Instance "cgms", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
09-20-2012 14:25:30 PDT: INFO: ==========  Completed Successfully ==========
```

Load-Balancing Policies

Table 1 describes the load-balancing policy for each type of traffic the LB supports:

Table 1

Traffic	Load Balancing Policy
HTTPS traffic to and from browsers and IoT FND API clients (IPv4; ports 80 and 443)	The LB uses Layer 7 load balancing for all traffic from Web browsers and IoT FND API clients. The LB uses stickiness for general HTTPS traffic.
For FAR IPv4 traffic going to ports 9121 and 9120: ■ Tunnel Provisioning on port 9120 over HTTPS ■ Regular registration and periodic on 9121 over HTTPS	The LB uses Layer 3 load balancing for all FAR traffic. This is the traffic from the FAR to IoT FND.
For IPv6 CSMP traffic to and from mesh endpoints (MEs): ■ UDP traffic over port 61624 – Registration – Periodic transmission of metrics – Firmware push – Configuration push ■ UDP traffic over port 61625 For outage notifications sent by MEs.	The LB uses Layer 3 load balancing for all ME traffic to port 61624, and outage messages to port 61625.

Running LB Configuration Example

The following is an example of the running configuration of a properly configured IoT FND LB:

```
# show running-config
Generating configuration....

ssh maxsessions 10

boot system image:c4710ace-t1k9-mz.A5_1_1.bin

hostname cgnmslb2
interface gigabitEthernet 1/1
  switchport access vlan 10
  no shutdown
interface gigabitEthernet 1/2
  description server-side
  switchport access vlan 11
  no shutdown
interface gigabitEthernet 1/3
  description client-side
  switchport access vlan 8
  no shutdown
interface gigabitEthernet 1/4
  switchport access vlan 55
  no shutdown
```

```
access-list ALL line 8 extended permit ip any any
access-list everyone line 8 extended permit ip any any
access-list everyone line 16 extended permit icmp any any
access-list ipv6_acl line 8 extended permit ip anyv6 anyv6
access-list ipv6_acl2 line 8 extended permit icmpv6 anyv6 anyv6



ip domain-lookup
ip domain-name cisco.com
ip name-server 171.68.226.120
ip name-server 171.70.168.183

probe http probe_cgnms-http
  port 80
  interval 15
  passdetect interval 60
  expect status 200 200
  open 1

rserver host 12-12-1-31
  ip address 12.12.1.31
  conn-limit max 4000000 min 4000000
  probe probe_cgnms-http
  inservice
rserver host 12-12-1-32
  ip address 12.12.1.32
  conn-limit max 4000000 min 4000000
  probe probe_cgnms-http
  inservice
rserver host 2002-cafe-server-202
  description realserver 2002:cafe:server::202
  ip address 2002::202
  conn-limit max 4000000 min 4000000
  probe probe_cgnms-http
  inservice
rserver host 2002-cafe-server-211
  ip address 2002:cafe:server::211
  conn-limit max 4000000 min 4000000
  probe probe_cgnms-http
  inservice

serverfarm host cgnms_2
  description cgnms-serverfarm
  probe probe_cgnms-http
  rserver 2002-cafe-server-202 61624
    conn-limit max 4000000 min 4000000
    inservice
  rserver 2002-cafe-server-211 61624
    conn-limit max 4000000 min 4000000
    inservice
serverfarm host cgnms_2_ipv4
  probe probe_cgnms-http
  rserver 12-12-1-31
    conn-limit max 4000000 min 4000000
    inservice
  rserver 12-12-1-32
    conn-limit max 4000000 min 4000000
    inservice

sticky ip-netmask 255.255.255.255 address source CGNMS_SRC_STICKY
  serverfarm cgnms_2_ipv4
```

```
class-map type management match-any remote_access
  2 match protocol xml-https any
  3 match protocol icmp any
  4 match protocol telnet any
  5 match protocol ssh any
  6 match protocol http any
  7 match protocol https any
  8 match protocol snmp any
class-map type management match-all ssh_allow_access
  2 match protocol ssh any
class-map match-any virtual-server-cgnms
  2 match virtual-address 2002:server:cafe::210 udp eq 61624
class-map match-any vs_cgnms_ipv4
  3 match virtual-address 12.12.1.101 tcp eq https
  4 match virtual-address 12.12.1.101 tcp eq 9120
  5 match virtual-address 12.12.1.101 tcp eq 9121
  6 match virtual-address 12.12.1.101 tcp eq 8443
  7 match virtual-address 12.12.1.101 tcp any

policy-map type management first-match remote_mgmt_allow_policy
  class remote_access
    permit

policy-map type loadbalance first-match virtual_cgnms_l7
  class class-default
    serverfarm cgnms_2
policy-map type loadbalance first-match vs_cgnms_l7_v4
  class class-default
    sticky-serverfarm CGNMS_SRC_STICKY

policy-map multi-match cgnms_policy_ipv6
  class virtual-server-cgnms
    loadbalance vip inservice
    loadbalance policy virtual_cgnms_l7
    loadbalance vip icmp-reply active
policy-map multi-match int1000
  class vs_cgnms_ipv4
    loadbalance vip inservice
    loadbalance policy vs_cgnms_l7_v4
    loadbalance vip icmp-reply active

interface vlan 8
  bridge-group 1
  access-group input everyone
  access-group input ipv6_acl
  no shutdown
interface vlan 10
  bridge-group 2
  access-group input everyone
  access-group input ipv6_acl
  service-policy input int1000
  no shutdown
interface vlan 11
  bridge-group 2
  access-group input everyone
  access-group input ipv6_acl
  no shutdown
interface vlan 55
  bridge-group 1
  access-group input everyone
  access-group input ipv6_acl
  service-policy input cgnms_policy_ipv6
```

```
  no shutdown

interface bvi 1
  ipv6 enable
  ip address 2002:server:cafe::206/64
  no shutdown
interface bvi 2
  ip address 12.12.1.100 255.255.255.0
  no shutdown

domain cisco.com

ip route 2011::/16 2002:server:cafe::101
ip route 2001:server:cafe::/64 2002:cafe::101
ip route 11.1.0.0 255.255.0.0 12.12.1.33
ip route 15.1.0.0 255.255.0.0 12.12.1.33
ip route 13.211.0.0 255.255.0.0 12.12.1.33

context VC_Setup1
  allocate-interface vlan 40
  allocate-interface vlan 50
  allocate-interface vlan 1000

username admin password 5 $1$CB34uAB9$BW8a3ijjxvBGttuGtTcST/  role Admin domain
default-domain
username www password 5 $1$q/YDKDp4$9PkZl1SBMQW7yZ7E.sOZA/  role Admin domain de
fault-domain

ssh key rsa 1024 force
```

Configuring Tunnel Provisioning Policies

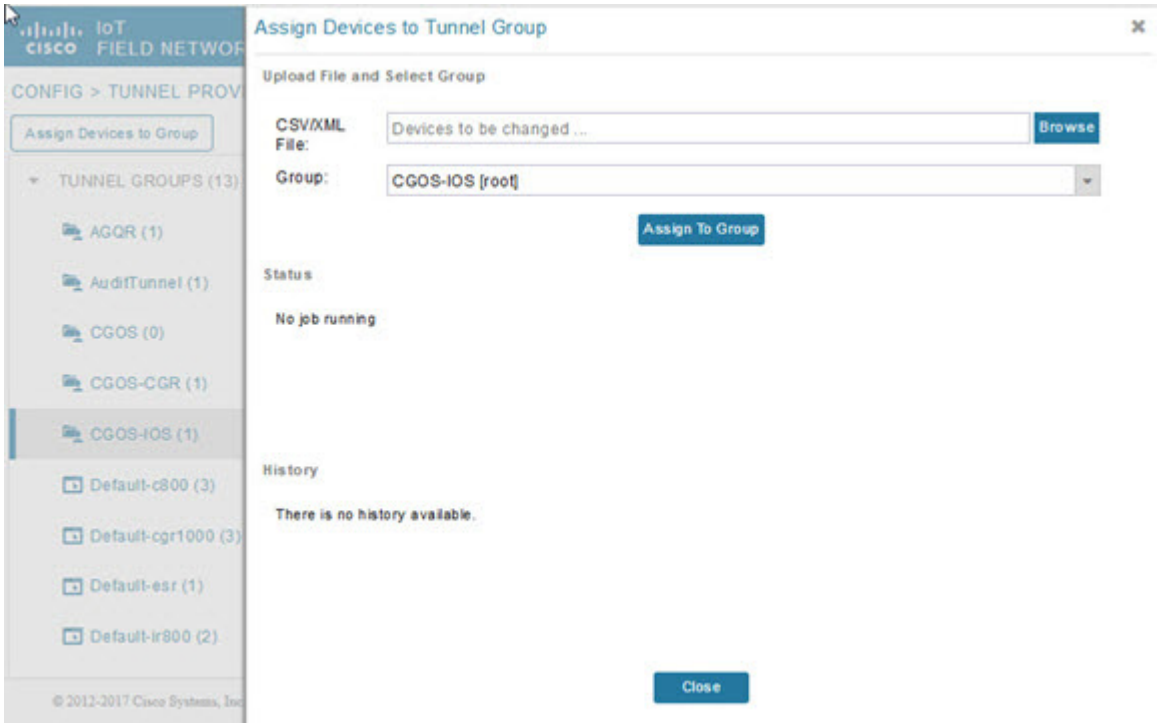
Use tunnel policies to configure multiple tunnels for a FAR. Each tunnel is associated with an interface on a FAR and an HER. If a tunnel provisioning group has one or more HERs, IoT FND displays a policy in the Tunnel Provisioning Policies tab (**Config > Tunnel Provisioning**). Use this policy to configure FAR-to-HER interface mapping.

To map FAR-to-HER interfaces in IoT FND:

1. Choose **CONFIG > Tunnel Provisioning**.
2. In the TUNNEL GROUPS pane, select a group to configure with tunnel redundancy.
3. Create a CSV or XML file that lists the HERs to add to the group in the format *EID, device type*, as follows:

   ```
   eid,deviceType
   asr-0, asr1000
   asr-1, asr1000
   asr-2, asr1000
   ```

4. Click **Assign Devices to Group** to import the file and add HERs to the group.



Note: A HER can be a member of multiple tunnel provisioning groups.

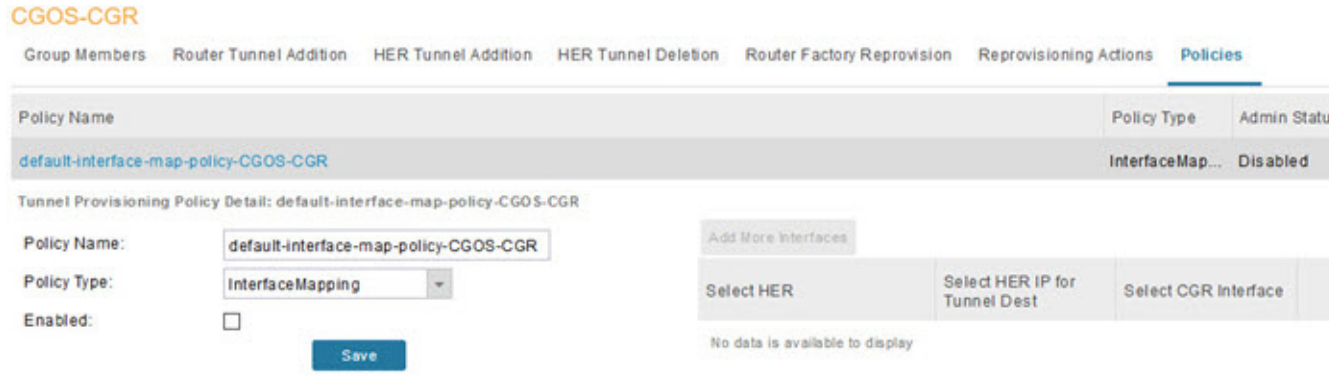
5. With the tunnel provisioning group selected, click the **Policies** tab.

   By default, IoT FND displays the **default-interface-mapping-policy-*tunnel-group*** name for the selected tunnel group within the Policy Name panel.

   Note: Interface-mapping is the only policy type currently supported in IoT FND.

   IoT FND displays one interface mapping entry for every HER in the group. You can add or remove interface mapping entries as needed.

6. Click the Policy Name link within the Policy Name Panel to open an entry panel. In the Policy Name field, enter the name of the policy.



To **add** an interface-mapping entry to the policy, click **Add More Interfaces** (button found above Select HER listing right-side of page). To **delete** an entry, click **Delete** (**X**) for that entry.

7. To configure an interface-mapping entry, click the Policy Name link, and complete the following as necessary:
   a. To select a different HER, click the currently selected HER and choose a different one from the **Select a HER** drop-down menu.
   b. To select the HER IP for the tunnel destination on the HER, click the selected interface and choose a different one from the **Select HER IP** drop-down menu.
   c. To select the FAR interface that maps to the selected HER interface, choose an interface from the **Select CGR Interface** drop-down menu.
   d. Click **Update**.
8. To enable the policy, check the **Enabled** check box.
9. Click **Save**.

Modifying the Tunnel Provisioning Templates for Tunnel Redundancy

After defining the tunnel provisioning policy for a tunnel provisioning group, modify the Field Area Router Tunnel Addition and the Head-End Router Tunnel Addition templates to include commands to establish the multiple tunnels defined in the policy.

Field Area Router Tunnel Addition Template Example

In this example, bold text indicates the changes made to the default Field Area Router Tunnel Addition template to create multiple tunnels:

```
<#--
  Configure a Loopback0 interface for the FAR. This is done first as features
  look for this interface and use it as a source.

  This is independent of policies
-->
interface Loopback0
  <#--
    Now obtain an IPv4 address that can be used to for this FAR's Loopback
    interface. The template API provides methods for requesting a lease from
    a DHCP server. The IPv4 address method requires a DHCP client ID and a link
    address to send in the DHCP request. The 3rd parameter is optional and
    defaults to "CG-NMS". This value is sent in the DHCP user class option.
    The API also provides the method "dhcpClientId". This method takes a DHCPv6
    Identity association identifier (IAID) and a DHCP Unique  IDentifier (DUID)
    and generates a DHCPv4 client identifier as specified in RFC 4361. This
    provides some consistency in how network elements are identified by the
    DHCP server.
  -->
  ip address ${far.ipv4Address(dhcpClientId(far.enDuid, 0), far.dhcpV4LoopbackLink).address}/32
  <#--
    Now obtain an IPv6 address that can be used to for this FAR's loopback
    interface. The method is similar to the one used for IPv4, except clients
    in DHCPv6 are directly identified by their DUID and IAID. IAIDs used for
    IPv4 are separate from IAIDs used for IPv6, so we can use zero for both
    requests.
  -->
  ipv6 address ${far.ipv6Address(far.enDuid, 0, far.dhcpV6LoopbackLink).address}/128
exit

<#-- Make certain the required features are enabled on the FAR. -->
feature crypto ike
feature ospf
feature ospfv3
```

```
feature tunnel
<#-- Features ike and tunnel must be enabled before ipsec. -->
feature crypto ipsec virtual-tunnel

<#--
  Toggle on/off the c1222r feature to be certain it uses the Loopback0
  interface as its source IP.
-->
no feature c1222r
feature c1222r

<#-- Configure Open Shortest Path First routing processes for IPv4 and IPv6. -->
router ospf 1
exit
router ospfv3 2
exit

<#--
  Now that OSPF has been configured complete the configuration of Loopback0.
-->
interface Loopback0
  ip router ospf 1 area ${far.ospfArea1!"1"}
  ipv6 router ospfv3 2 area ${far.ospfV3Area1!"0"}
exit

<#-- Configure Internet Key Exchange for use by the IPsec tunnel(s). -->
crypto ike domain ipsec
  identity hostname
  policy 1
    <#-- Use RSA signatures for the authentication method. -->
    authentication rsa-sig
    <#-- Use the 1536-bit modular exponential group. -->
    group 5
  exit
exit
crypto ipsec transform-set IPSecTransformSet esp-aes 128 esp-sha1-hmac
crypto ipsec profile IPSecProfile
  set transform-set IPSecTransformSet
exit

<#--
  Define template variables to keep track of the next available IAID (IPv4)
  and the next available tunnel interface number. We used zero when leasing
  addresses for Loopback0, so start the IAID at one.
-->
<#assign iaId = 1>
<#assign interfaceNumber = 0>

<#--
  The same logic is needed for each of the IPsec tunnels, so a macro is used
  to avoid duplicating configuration. The first parameter is the prefix to
  use when looking for the WAN interface on the FAR to use for the source of
  the tunnel. The second parameter is the OSPF cost to assign to the tunnel.
-->
<#macro configureTunnel interfaceNamePrefix destinationInterface her tunnelIndex ospfCost>
  <#--
    If an interface exists on the FAR whose name starts with the given prefix
    and an IPv4 address as been assigned to that interface then the IPsec
    tunnel can be configured, otherwise no tunnel will be configured. The
    template API interfaces method will return all interfaces whose name
    starts with the given prefix.
  -->
  <#assign wanInterface = far.interfaces(interfaceNamePrefix)>
```

```
  <#-- Check if an interface was found and it has an IPv4 address. -->
  <#if (wanInterface[0].v4.addresses[0].address)??>
    <#--
      Determine the HER destination address to use when configuring the tunnel.
      If the optional property "ipsecTunnelDestAddr1" has been set on this FAR
      then use the value of that property. Otherwise look for that same property
      on the HER. If the property is not set on the FAR or the HER, then fallback
      to using an address on the HER GigabitEthernet0/0/0 interface.
    -->
    <#assign destinationAddress = her.interfaces(destinationInterface)[0].v4.addresses[0].address>

    <#if !(destinationAddress??)>
      ${provisioningFailed("Unable to determine the destination address for IPsec tunnels")}
    </#if>
    interface Tunnel${interfaceNumber}
      <#assign interfaceNumber = interfaceNumber + 1>
      description IPsec tunnel to ${her.eid}
      <#--
        For a tunnel interface two addresses in their own tiny subnet are
        needed. The template API provides an ipv4Subnet method for leasing an
        IPv4 from a DHCP server. The parameters match those of ipv4Address,
        with a fourth optional parameter that can be used to specify the
        prefix length of the subnet to request. If not specified the prefix
        length requested will default to 31, which provides the two addresses
        needed for a point to point link.

        NOTE: If the DHCP server being used does not support leasing an IPv4
        subnet, then this call will have to be changed to use the ipv4Address
        method and the DHCP server will have to be configured to respond
        appropriately to the request made here and the second request that
        will have to be made when configuring the HER side of the tunnel.
        That may require configuring the DHCP server with reserved addresses
        for the client identifiers used in the calls.
      -->
      <#assign lease = far.ipv4Subnet(dhcpClientId(far.enDuid, tunnelIndex), far.dhcpV4TunnelLink)>
      <#assign iaId = iaId + 1>
      <#-- Use the second address in the subnet for this side of the tunnel. -->
      ip address ${lease.secondAddress}/${lease.prefixLength}
      ip ospf cost ${ospfCost}
      ip ospf mtu-ignore
      ip router ospf 1 area ${far.ospfArea1!"1"}
      tunnel destination ${destinationAddress}
      tunnel mode ipsec ipv4
      tunnel protection ipsec profile IPSecProfile
      tunnel source ${wanInterface[0].name}
      no shutdown
    exit
  </#if>
</#macro>

<#--
  Since we are doing policies for each tunnel here, the list of policies passed to this template can be
  iterated over to get the tunnel configuration viz interface mapping

  tunnelObject.ipSecTunnelDestInterface is the "interface on CGR"
  tunnelObject.ipSecTunnelSrcInterface is the "interface on HER"
  tunnelObject.her is the HER of interest
 -->

<#list far.tunnels("ipSec") as tunnelObject>
    <@configureTunnel tunnelObject.ipSecTunnelDestInterface tunnelObject.ipSecTunnelSrcInterface
tunnelObject.her tunnelObject.tunnelIndex 100/>  <----- Loop through policies (aka Tunnels)
</#list>

<#--
```

```
  Make certain provisioning fails if we were unable to configure any IPsec
  tunnels. For example this could happen if the interface properties are
  set incorrectly.
-->
<#if iaId = 1>
  ${provisioningFailed("Did not find any WAN interfaces to use as the source for IPsec tunnels")}
</#if>

<#--
  Configure an IPv6-in-IPv4 GRE tunnel to allow IPv6 traffic to reach the data
  center.
-->
<#macro configureGreTunnel destinationInterface her tunnelIndex>

<#assign destinationAddress = her.interfaces(destinationInterface)[0].v4.addresses[0].address>

<#if !(destinationAddress??)>
     ${provisioningFailed("Unable to determine the destination address for GRE tunnels")}
</#if>

interface Tunnel${interfaceNumber}
  <#assign interfaceNumber = interfaceNumber + 1>
  description GRE IPv6 tunnel to ${her.eid}
  <#--
    The ipv6Subnet method is similar to the ipv4Subnet method except instead
    of obtaining an IPv4 subnet it uses DHCPv6 prefix delegation to obtain an
    IPv6 prefix. The prefix length will default to 127, providing the two
    addresses needed for the point to point link. For the IAID, zero was used
    when requesting an IPv6 address for loopback0, so use one in this request.
  -->
  <#assign lease = far.ipv6Subnet(far.enDuid, tunnelIndex, far.dhcpV6TunnelLink)>
  ipv6 address ${lease.secondAddress}/${lease.prefixLength}
  ipv6 router ospfv3 2 area ${far.ospfV3Area1!"0"}
  ospfv3 mtu-ignore
  tunnel destination ${destinationAddress}
  tunnel mode gre ip
  tunnel source Loopback0
  no shutdown
exit

</#macro>

<#-- Loop through the policies for GRE tunnels -->
<#list far.tunnels("gre") as greTunnelObj>
    <@configureGreTunnel greTunnelObj.greDestInterface greTunnelObj.her greTunnelObj.tunnelIndex/>
</#list>
```

Head-End Router Tunnel Addition Template

In this example, bold text indicates the changes made to the default Head-End Router Tunnel Addition template to create multiple tunnels:

```
<#--
  Define template variables to keep track of the IAID (IPv4) that was used by
  the FAR template when configuring the other end of the tunnel. This template
  must use the same IAID in order to locate the same subnet that was leased by
  the FAR template so both endpoints are in the matching subnet.
-->
<#assign iaId = 1>

<#--
```

```
  The same logic is needed for each of the IPsec tunnels, so a macro is used.
-->
<#macro configureTunnel ipSecTunnelSrcInterface ipSecTunnelDestInterface her tunnelIndex ospfCost>
  <#--
    Only configure the HER tunnel end point if the FAR tunnel end point was
    configured. This must match the corresponding logic in the FAR tunnel
    template. The tunnel will not have been configured if the WAN interface
    does not exist on the FAR or does not have an address assigned to it.
  -->
  <#assign wanInterface = far.interfaces(ipSecTunnelDestInterface)>
  <#if (wanInterface[0].v4.addresses[0].address)??>
    <#-- Obtain the full interface name based on the prefix. -->
    <#assign interfaceName = wanInterface[0].name>
    <#--
      Locate a tunnel interface on the HER that is not in use. The template
      API provides an unusedInterfaceNumber method for this purpose. All of
      the parameters are optional. The first parameter is a name prefix
      identifying the type of interfaces, it defaults to "tunnel". The second
      parameter is a lower bound on the range the unused interface number must
      be in, it defaults to zero. The third parameter is the upper bound on
      the range, it defaults to max integer (signed). The method remembers
      the unused interface numbers it has returned while the template is
      being processed and excludes previously returned numbers. If no unused
      interface number meets the constraints an exception will be thrown.
    -->
    interface Tunnel${her.unusedInterfaceNumber()}
      description IPsec tunnel to ${far.eid}
      <#assign lease = far.ipv4Subnet(dhcpClientId(far.enDuid, tunnelIndex), far.dhcpV4TunnelLink)>
      <#assign iaId = iaId + 1>
      ip address ${lease.firstAddress} ${lease.subnetMask}
      ip ospf cost ${ospfCost}
      ip ospf mtu-ignore
      tunnel destination ${wanInterface[0].v4.addresses[0].address}
      tunnel mode ipsec ipv4
      tunnel protection ipsec profile IPSecProfile
      tunnel source ${ipSecTunnelSrcInterface}
      no shutdown
    exit
    router ospf 1
      network ${lease.prefix} ${lease.wildcardMask} area ${far.ospfArea1!"1"}
    exit
  </#if>
</#macro>

<#list far.tunnels("ipSec") as tunnelObject>
        <@configureTunnel tunnelObject.ipSecTunnelSrcInterface tunnelObject.ipSecTunnelDestInterface
tunnelObject.her tunnelObject.tunnelIndex 100/>
</#list>

<#--
  Configure an IPv6-in-IPv4 GRE tunnel to allow IPv6 traffic to reach the data
  center.
-->
<#macro configureGreTunnel greSrcInterface her tunnelIndex>
interface Tunnel${her.unusedInterfaceNumber()}
  description GRE IPv6 tunnel to ${far.eid}
  <#assign lease = far.ipv6Subnet(far.enDuid, tunnelIndex, far.dhcpV6TunnelLink)>
  ipv6 address ${lease.firstAddress}/${lease.prefixLength}
  ipv6 enable
  ipv6 ospf 2 area ${far.ospfV3Area1!"0"}
  ipv6 ospf mtu-ignore
  tunnel destination ${far.interfaces("Loopback0")[0].v4.addresses[0].address}
  tunnel mode gre ip
  tunnel source ${greSrcInterface}
exit
```

```
</#macro>

<#-- Loop through the policies for GRE tunnels -->
<#list far.tunnels("gre") as greTunnelObj>
        <@configureGreTunnel greTunnelObj.greSrcInterface greTunnelObj.her greTunnelObj.tunnelIndex/>
</#list>
```