ORACLE

Fast Application Notification (FAN)

Includes FANwatcher
A utility to subscribe to ONS and view FAN events

May, 2025, Version [1.0]
Copyright © 2025, Oracle and/or its affiliates
Public

Purpose statement

This document provides an overview of features and enhancements included in release 23ai. It is intended solely to help you assess the business benefits of upgrading to 23ai and planning for the implementation and upgrade of the product features described.

Disclaimer

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

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

Copyright © 2025, Oracle and/or its affiliates / Public

ORACLE

Table of contents

Purpose and Audience **5**

Benefits of Using FAN Events **6**

Why is it important to use FAN? 6

FAN with Oracle Database 23ai 6

How to use FAN 7

How do you use FAN for Fast Failover 8

How do you use FAN for Transparent Planned Maintenance 9

What do FAN events look like? **10**

FAN High Availability Events 10

FAN Load Balancing Advisory Events 13

Viewing FAN events **14**

FANwatcher 14

Server-side Fast Application Notification Callouts 18

Configuring Applications to use FAN with Fast Connection Failover **22**

How to Configure the Oracle Notification Service for 23ai Grid Infrastructure 23

General Steps for Configuring FCF Clients 24

How to Configure FAN for 23ai Java Clients 27

How to Configure FAN for ODP.Net Clients, Managed and Unmanaged Providers 29

How to Configure FAN for OCI Clients 31

Conclusion **35**

Appendix A: Configuring ONS **36**

ONS Configuration File 36

Client-side ONS Configuration 38

Remote ONS Configuration 41

Appendix B Troubleshooting FAN **43**

Copyright © 2025, Oracle and/or its affiliates / Public

ORACLE

List of figures

Figure 1: Timeline showing FAN activity during Instance Failure 8

Figure 2: Timeline showing FAN activity during Planned Maintenance 9

List of tables

Table 1: FAN Connection Signature 10

Table 2: FAN High Availability Event Types 11

Table 3: FAN High Availability Event Status 11

Table 4: FAN High Availability Event Reasons 11

Table 5: FAN High Availability Event Payload Fields 12

Table 6: FAN Runtime Load Balancing Advisory Event 14

Table 7: Example filtering criteria for specific FAN event handlers 19

Table 8. Fast Connection Failover by Oracle Database Release 23

Table 9. ons.config Parameters and Definition 36

Table 10. onsctl commands, action, and output 38

List of figures and tables

Figure 1: Timeline showing FAN activity during Instance Failure 8

Figure 2: Timeline showing FAN activity during Planned Maintenance 9

Table 1: FAN Connection Signature 10

Table 2: FAN High Availability Event Types 11

Table 3: FAN High Availability Event Status 11

Table 4: FAN High Availability Event Reasons 11

Table 5: FAN High Availability Event Payload Fields 12

Table 6: FAN Runtime Load Balancing Advisory Event 14

Table 7: Example filtering criteria for specific FAN event handlers 19

Table 8. Fast Connection Failover by Oracle Database Release 23

Table 9. ons.config Parameters and Definition 36

Table 10. onsctl commands, action, and output 38

Copyright © 2025, Oracle and/or its affiliates / Public

Purpose and Audience

This technical brief discusses Oracle 23*ai Fast Application Notification* (FAN)[1] in Oracle Real Application Clusters (RAC) and Data Guard environments. FAN is a critical component towards solving the poor experience that end users can encounter when planned maintenance, unplanned outages, and load imbalances make database instances unavailable or unresponsive. FAN enables end-to-end, lights-out recovery of applications and load balancing at runtime based on real transaction performance.

FAN publishes the database's state to applications. When the state of database services changes (for example, up, down, or unresponsive), the new status is posted to interested subscribers through FAN events. FAN provides rapid notification about state changes for database services, instances, databases, and cluster nodes.

Oracle drivers and Oracle pools use FAN events to achieve the following:

- Draining of work during planned maintenance, with no errors returned to the application.
- Very fast detection of failures so that recovery of applications can occur in real-time
- Load balancing of incoming work at runtime when performance imbalances occur, following instances leaving and joining the system, and resources becoming available.
- Affinity advice for incoming work so that related conversations, such as successive web sessions, are routed together for the best performance.

This paper:

1. Outlines the benefits of enabling FAN events for your database system
2. Dissects the FAN events and their published fields
3. Describes how to view FAN events using FANwatcher
4. Steps to integrate and enable FAN events

The target audience includes RAC database administrators, Data Guard database administrators, and application integrators who need rapid notification of planned maintenance and unplanned outages integrated with their applications or application servers, monitoring consoles, or internal business workflow systems. It is assumed that the reader is familiar with the concepts presented in the following references:

Oracle Clusterware Administration and Deployment Guide 23ai Part number F46761-03

Oracle Real Application Clusters Administration and Deployment Guide 23ai Part Number F46762-03

[1] This paper is relevant to all releases since 19c. Some information may be specific to Oracle Database 23ai (command syntax, for example).

Copyright © 2025, Oracle and/or its affiliates / Public

ORACLE

Benefits of Using FAN Events

Why is it essential to use FAN?

State changes during planned and unplanned outages impact application response time:

- Waiting for TCP/IP time-outs for minutes when a node fails without closing sockets, and for every subsequent connection while that IP address is down.
- Attempting to connect when services are down.
- Not connecting when services resume.
- Processing the last result at the client when the server goes down.
- Attempting to execute work on slow, hung, and dead nodes.

Still worse, these sources of poor performance are not captured and reported by the standard database performance methodologies. When a node fails without closing sockets, all sessions blocked in an IO (read or write) wait until TCP keepalive. This wait status is the typical condition for an application using databases. Sessions processing the last result are even worse, as they will not receive an interrupt until the next data is requested. Using FAN events eliminates applications waiting on TCP time-outs, time wasted processing the client's last result after a failure, and time wasted executing work on slow, hung, or dead nodes.

Typically, client or mid-tier applications connected to the database relied on connection timeouts, out-of-band polling mechanisms, or other custom solutions to realize that a system component had failed, triggering actions to mitigate the impact of that failure. This approach had severe implications in (1) application availability because of extended downtimes and noticeable (due to polling intervals and timeout length) and (2) management of enterprise environments because of the exponential growth in the number of server-side components, both horizontally (across all nodes) and vertically (across node components such as listeners, instances and application services) caused a significant increase in the sheer number of polling actions.

FAN, released with Oracle Database 10g, solved these problems by integrating database events with listeners, applications, application servers, and IT management consoles (including Oracle Enterprise Manager, trouble ticket loggers, and e-mail/texting servers). FAN events are posted as soon as a system change is detected, resulting in immediate action at the incident and overall improved resource usage as applications do not need to implement unnecessary status polling.

FAN with Oracle Database 23ai

Starting with Oracle Database 12c, there are three essential enhancements for FAN –

- FAN is default, configured, and enabled out of the box with Oracle RAC
- All Oracle clients use the Oracle Notification System (ONS) as the transport for FAN
- FAN is posted by Global Data Services (GDS) to allow FAN events to span data centers

FAN is on by Default

By default, FAN is configured during Oracle Grid Infrastructure installation. `srvctl` can be used to modify the ports used by FAN. Oracle clients do not need any configuration changes to receive FAN events. This means that when the client starts, it automatically queries the databases for the ONS end-points. The automatic configuration spans data centers. The client automatically receives an ONS configuration from each database in the connection URL. No configuration of ONS is required at the client other than enabling FAN.

FAN standardizes on ONS as the Transport

Starting with Oracle Database 12c, all Oracle 12c FAN clients use the Oracle Notification System to receive FAN events. The reason for standardizing on ONS as the transport for FAN is that “the failed component cannot tell you that it has failed”. ONS runs in the Grid Infrastructure outside of the database stack and with any database dependencies. When a database is stopped or fails, FAN posts the status change events immediately, and ONS

Copyright © 2025, Oracle and/or its affiliates / Public

delivers them immediately. To support clients from earlier releases and new clients using older database versions, events are automatically published over both transport methods, ONS and AQ.

Global Data Services (GDS)

From Oracle Database 12c, Global Data Services (GDS) posts FAN for spanning data centers, particularly for Oracle Active Data Guard farms that need runtime load balancing. GDS considers the service placement attributes, automatically performs an inter-database service failover to another available database for planned maintenance that involves a data center change and notifies failures of an entire database if unplanned outages occur. The Oracle clients and connection pools are interrupted on failure events and notified when a global service has been newly started.

How to use FAN

Using FAN requires no code changes. The best and easiest way to use FAN is to use an Oracle connection pool, an Oracle client driver, or an Oracle Application configured for FAN.

FAN events are integrated with:

- Oracle Fusion Middleware and Oracle WebLogic Server (Continuous Availability for Oracle WebLogic Server Part Number E90842-03)
- Oracle Data Guard Broker *(Oracle® Data Guard Broker 23ai* Part Number F46805-02)
- Oracle Enterprise Manager Cloud Control (Basic Installation Guide 24ai Release 1 (24.1) Part Number F97194-03)
- Oracle JDBC Universal Connection Pool (*Oracle® Universal Connection Pool Developer's Guide 23ai* Part Number F47026-10)
- ODP.NET (Oracle® Database Development Guide 23ai Part Number F47572-16)
- SQLPLUS (SQL*Plus® User's Guide and Reference 23ai Part Number F47057-07)
- Global Data Services (Oracle® Database Global Data Services Concepts and Administration Guide 23ai Part Number F46808-03)

When using third-party clients such as IBM WebSphere and Apache Tomcat, FAN is supported by IBM and Apache, respectively, by replacing the default connection pool with the Oracle Universal Connection Pool. Oracle's Universal Connection Pool (UCP) can hide planned maintenance when used with FAN and unplanned outages with Oracle Database 23ai and Application Continuity. UCP is a proven and recommended client solution for Java-based applications that use Oracle RAC, Oracle Data Guard, or Active Data Guard.

Copyright © 2025, Oracle and/or its affiliates / Public

How do you use FAN for Fast Failover

Figure 1: Timeline showing FAN activity during Instance Failure

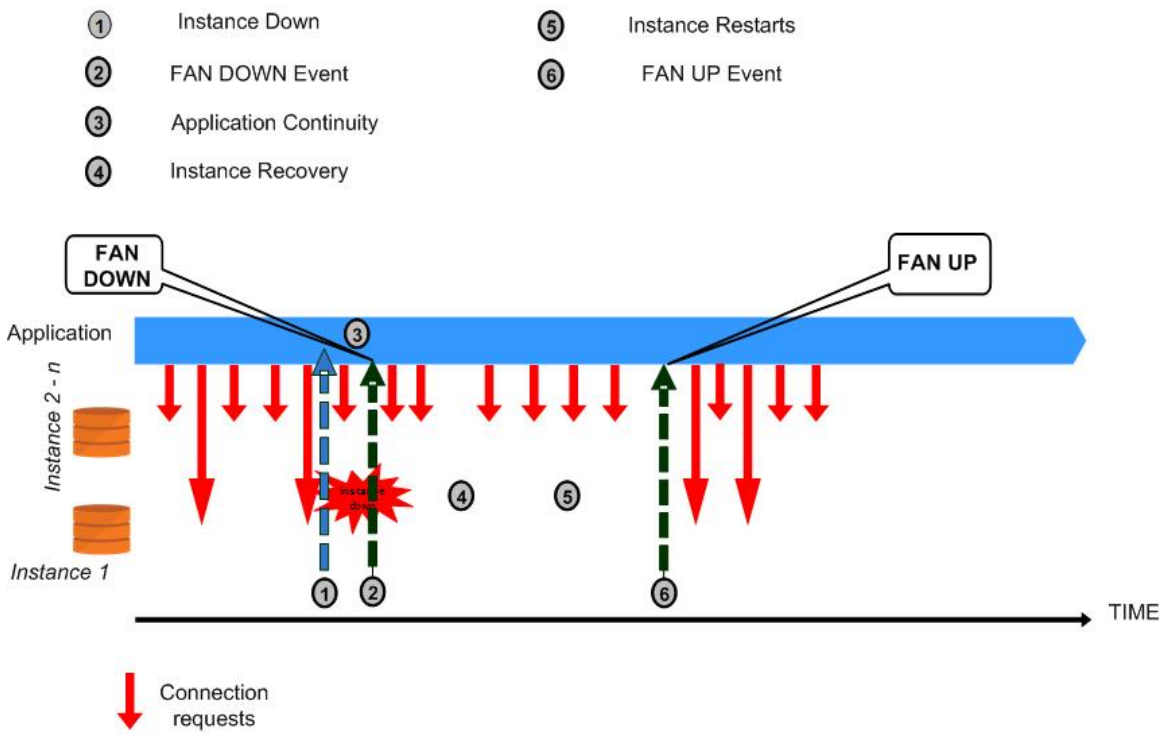

Figure 1 shows a timeline for Instance Failure. The application is FAN-enabled (UCP with Fast Connection Failover enabled) and connects using a dynamic database service running on all instances of an Oracle RAC database or in a configuration where the service runs on a subset of instances. The application has active sessions in all instances. The following occurs during unplanned outages. This example uses instance failure:

- In Step 1: The instance crashes.
- In Step 2: The FAN unplanned DOWN event immediately clears idle sessions from the connection pool.

Existing connections on other instances remain usable, and new connections are opened to these instances if needed.

- For pools configured to use Application Continuity, active sessions are restored to a surviving instance and recovered by Application Continuity, masking the outage from users and applications.
  If not protected by Application Continuity, any sessions in active communication with the instance will receive an error, as shown in Step 3. The application can test if this is a recoverable error and could initiate a reconnection attempt at this time. If, as in our example, the service is available on another instance, they could get a new connection.
- As this is a RAC database, the surviving instance will perform recovery for the failed instance, as shown in Step 6, and then the instance is restarted by the Grid Infrastructure (Step 7).
- The service's FAN UP event informs the connection pool that a new instance is available for use, allowing sessions to be created on this instance at the next request submission.

Copyright © 2025, Oracle and/or its affiliates / Public

How do you use FAN for Transparent Planned Maintenance

Figure 2: Timeline showing FAN activity during Planned Maintenance

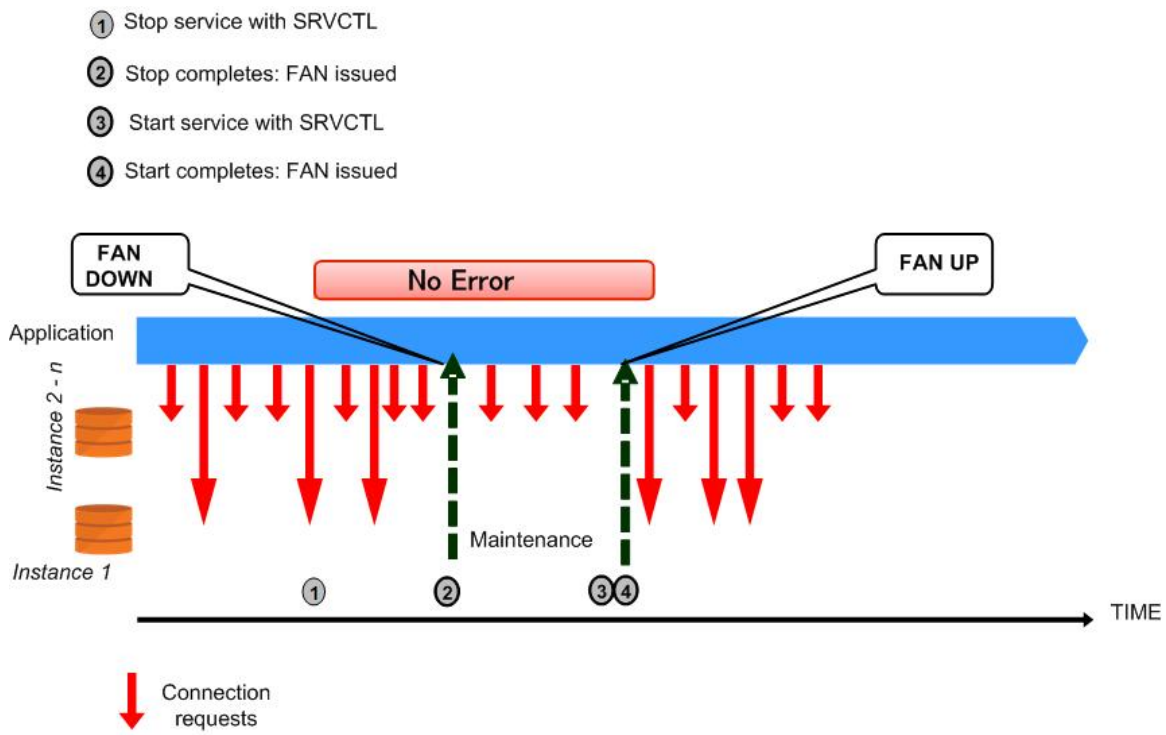

Figure 2 shows a timeline for planned maintenance, such as applying a database patch. The application is FAN-enabled (UCP with Fast Connection Failover enabled) and connects using a dynamic database service running on all instances of an Oracle RAC database or in a configuration where the service runs on a subset of instances. The application has active sessions in all instances. The application is uninterrupted during this planned outage as the patch is applied in a rolling fashion:

- In Step 1: Use `srvctl` to stop the service configured as a UNIFORM service on an instance. If the service is not UNIFORM, relocate it to another instance. Do not use the `-force` flag with any of these commands. The connection pool automatically releases a connection at a request boundary, typically when the connection is checked into the pool.
- The FAN planned `DOWN` event, delivered at Step 2, clears idle sessions from the connection pool immediately and marks active sessions to be released and returned to the pool. These FAN actions drain the sessions from the instance without disrupting the application or users.

Existing connections on other instances remain usable, and new connections are opened to these instances if needed.

- Not all sessions, in all cases, will check their connections into the pool. It is best practice to have a timeout period after which the instance is forcibly shut down, evicting any remaining client connections.
  For those pools configured to use Application Continuity, Application Continuity recovers the remaining sessions, masking the outage from users and applications.
- Once the upgrade, patch, or repair is complete, restart the instance and the service on the original node using srvctl (Step 3).
- The service's FAN UP event informs the connection pool that a new instance is available for use, allowing sessions to be created on this instance at the next request submission.

Copyright © 2025, Oracle and/or its affiliates / Public

What do FAN events look like?

Oracle FAN events consist of header and payload information delivered as a set of name-value pairs describing the name, type, and nature of the event and the time that the event occurred. Based on this payload, the event recipient, normally an Oracle Client, takes actions such as refreshing stale connection references for Oracle connection pools, automatically recovering the in-flight work when that client is configured to use Application Continuity, determining the last committed outcome, and returning this to the user when that client is configured to use Transaction Guard, logging a service request or sending a text to the database administrator when that client is Enterprise Manager.

A FAN client uses the information on the connection, called the connection signature, to know which connections to operate on when an event arrives. (Table 1)

Table 1: FAN Connection Signature

FAN Parameter	Matching Database Signature
Service	`sys_context('userenv', 'service_name')`
Database unique name	`sys_context('userenv','db_unique_name')`
Instance	`sys_context('userenv', 'instance_name')`
Node name	`sys_context('userenv', 'server_host')`

FAN High Availability Events

Database connections using user-defined services automatically receive all FAN events. Database services decouple any hardwired mapping between a connection request and a database instance and provide value-added functionality, such as connection load balancing, dynamic workload management, service levels and priorities, connection pool reorganizations, and Application Continuity.

The event types that are important for Oracle FAN are:

- Service events for dynamic database services.
- Node events, which include cluster membership states and native join/leave operations for nodes
- Instance events that are used by OCI-based clients to group operations
- Connection pool clients use runtime load balancing and affinity advice to direct incoming database requests to the best place to run based on prior affinity and load advice.

FAN standardizes generating, presenting, and delivering events related to managed database resources for high availability and runtime load balancing. The node and VIP events require Oracle Grid Infrastructure. You can use Oracle RAC, Oracle RAC One Node, Oracle Restart, and Oracle Data Guard with Grid Infrastructure for all event types and GDS across data centers.

Copyright © 2025, Oracle and/or its affiliates / Public

Table 2: FAN High Availability Event Types

Event Type	Description
event_type=NODE	Oracle cluster node or network event
event_type=INSTANCE	Oracle Database Instance event
event_type=DATABASE	Oracle Database event
event_type=SERVICEMEMBER	Application service on a particular instance event
event_type=SERVICE	Application service event

Table 3: FAN High Availability Event Status

The *event status* for each of these managed resources includes:

Event Status	Description
status=up	Resource has started
status=down	Resource has stopped
status=nodedown	Node or public network has gone offline
status=not_restarting	Resource has failed more than 3 times in 30 minutes so is not being restarted

Table 4: FAN High Availability Event Reasons

The event status for each managed resource is associated with an *event reason*. The reason further describes what triggered the event:

Event Reason	Activity Type	Event triggered by
reason=USER	Planned	User-initiated commands by `srvctl, sqlplus, or gdsctl`
reason=FAILURE	Unplanned	A failure has been detected for that resource
reason=member_leave	Unplanned or Planned	A node has gone offline
reason=public_nw_down	Unplanned or Planned	Public network has gone offline
reason=BOOT	Unplanned	When a node starts, `DATABASE, INSTANCE` or `SERVICEMEMBER` events will start with `reason=BOOT` if they were online before the node left the cluster

Copyright © 2025, Oracle and/or its affiliates / Public

Table 5: FAN High Availability Event Payload Fields

Additional event payload fields further describe the unique resource whose status is being monitored and published. These additional fields include:

Event Resource Identifier	Description
VERSION=<n.n>	Event payload version (currently 1.0)
timestamp=<eventDate> <eventTime>	Server-side date and time when the event was detected
service=<serviceName.dbDomainName>	Fully qualified name of the database service
database=<dbName>	Database unique name
instance=<SID>	Name of the database instance
host=<hostname>	Name of the cluster node (as returned by Grid Infrastructure)
card=<n>	Service membership cardinality
timezone=<+|- GMT>	Difference from GMT
incarn	Cluster Incarnation number. Only visible in `NODE DOWN` event for guaranteed ordering
vip_ips	The IP address(es) of any VIP(s) that has/have gone down due to a public network failure. Only visible in Node DOWN events with `reason=public_nw_down`.

Example FAN High Availability Events

The result is a FAN event with one of the following payload structures:

Node Events

- Down

```
VERSION=1.0 event_type=NODE host=rachost_724 incarn=316152110 status=nodedown
reason=member_leave timestamp=2025-01-21 19:13:24 timezone=-08:00
```

- Public Network Down

```
VERSION=1.0 event_type=NODE host=rachost_724 incarn=0 status=nodedown
reason=public_nw_down vip_ips=10.10.8.249 timestamp=2025-01-21 19:13:13
timezone=-08:00
```

Note: `incarn` is always 0 when the `reason=public_nw_down`

Instance Events

- Instance Down

```
VERSION=1.0 event_type=INSTANCE service=testy.us.oracle.com instance=TESTY_2
database=testy db_domain=us.oracle.com host=rachost_723 status=down
reason=FAILURE timestamp=2025-01-21 19:32:43 timezone=-08:00
```

- Instance Up

```
VERSION=1.0 event_type=INSTANCE service=testy.us.oracle.com instance=TESTY_2
database=testy db_domain=us.oracle.com host=rachost_723 status=up reason=FAILURE
timestamp=2025-01-21 19:33:10 timezone=-08:00
```


Copyright © 2025, Oracle and/or its affiliates / Public

Service Events

- Servicemember Down

```
VERSION=1.0 event_type=SERVICEMEMBER service=testy_pdb_srv.us.oracle.com
instance=TESTY_2 database=testy db_domain=us.oracle.com host=rachost_723
status=down reason=FAILURE timestamp=2025-01-21 19:32:43 timezone=-08:00
```

- Servicemember Up

```
VERSION=1.0 event_type=SERVICEMEMBER service=testy_pdb_srv.us.oracle.com
instance=TESTY_2 database=testy db_domain=us.oracle.com host=rachost_723
status=up card=3 reason=BOOT timestamp=2025-01-21 19:33:15 timezone=-08:00
```

- Service Down

```
VERSION=1.0 event_type=SERVICE service=testy_pdb_srv.us.oracle.com
database=testy db_domain=us.oracle.com host=rachost_723 status=down reason=USER
timestamp=2025-01-21 19:25:52 timezone=-08:00
```

- Service Up

```
VERSION=1.0 event_type=SERVICE service=testy_pdb_srv.us.oracle.com
database=testy db_domain=us.oracle.com host=rachost_723 status=up reason=USER
timestamp=2025-01-22 17:57:13 timezone=-08:00
```

Database Events

- Database Down

```
VERSION=1.0 event_type=DATABASE service=testy.us.oracle.com database=testy
db_domain=us.oracle.com host=rachost_725 status=down reason=USER timestamp=2025-
03-24 20:09:17 timezone=-08:00
```

- Database Up

```
VERSION=1.0 event_type=DATABASE service=testy.us.oracle.com database=testy
db_domain=us.oracle.com host=rachost_725 status=up reason=USER timestamp=2025-
03-24 21:09:27 timezone=-08:00
```

FAN Load Balancing Advisory Events

Runtime Load Balancing events have been available since Oracle Database 10g Release 2. These events guide subscribers as to where to place incoming loads. By enabling runtime load balancing at the service level, FAN assists the application servers in directing work requests to where the request can be best satisfied based on the current response times, throughput, and capacity. Runtime load balancing works with affinity advice to keep requests in the same conversation together and move work away from slow, hung, and unresponsive nodes. Combined with affinity advice, runtime load balancing provides predictable response time or throughput across the system based on the service goal.

Runtime Load Balancing is provided out of the box for WebLogic Server Active GridLink, Oracle JDBC Universal Connection Pool, ODP.NET unmanaged and managed providers, PHP, and OCI Session Pool. All that is needed is to enable Runtime Load Balancing for the service, FAN, and, for ODP.NET, load balancing at the client. Load Balancing Advisory events are posted over ONS for all 23ai clients and over Advanced Queueing (AQ) for pre-12c OCI clients and all clients using pre-12c databases.

Copyright © 2025, Oracle and/or its affiliates / Public

Table 6: FAN Runtime Load Balancing Advisory Event

Event Resource Identifier	Description
Version	Version of the event payload.
Event type	SERVICE_METRICS
Service	Matches the service in DBA_SERVICES.
Database unique name	The unique database supporting the service. Matches the initialization parameter value for db_unique_name, which defaults to the value of the initialization parameter DB_NAME.
Timestamp	Date and time stamp (local time zone) to order events
Repeated for each event	
Instance	The name of the instance supporting the service. Matches the ORACLE_SID
Percent	The percentage of work requests to send to this database and instance
Service Quality	Weighted moving average of the service quality and bandwidth, based on the goal for the service (elapsed time or throughput)
Flag	Indication of the service quality relative to the service goal – Values are GOOD, VIOLATING, NO DATA, UNKNOWN Note: flag=UNKNOWN will occur if application work is not active on the instance (for the listed service) `flag=NO DATA` will display if the instance misses consecutive updates on load data (for the listed service)

Load Balancing Advisory Example

```
VERSION=1.0 database=TESTY service=testy_pdb_srv.us.oracle.com {

{instance=TESTY_2 percent=34 flag=UNKNOWN aff=FALSE}{instance=TESTY_3

percent=33 flag=UNKNOWN aff=FALSE}{instance=TESTY_1 percent=33 flag=UNKNOWN

aff=FALSE} } timestamp=2025-01-22 18:41:41
```

Viewing FAN events

FAN events are posted automatically when using the 23ai Grid Infrastructure. When using an Oracle Database 23ai client, there is no further configuration other than enabling FAN at the client.

As an administrator, you may wish to see the FAN events flowing. It's also valuable to keep a record of FAN high availability events posted by your system. FAN callouts can provide an audit trail and timing information for what events occurred and when.

FANwatcher

The `FANwatcher` utility is a self-contained Java application that connects to a local ONS (Oracle Notification Service) daemon and subscribes to FAN events. The utility displays a portion of the FAN event header information and the event payload.

The `FANwatcher` utility can run on an Oracle client installation, on any node in a Grid Infrastructure cluster, or on any mid-tier node that has an ONS daemon installed (for example, a node supporting WebLogic Server Active Gridlink for RAC).

The ONS daemon that is being subscribed to must be configured to communicate with the ONS daemon running as part of Grid Infrastructure, as the Grid Infrastructure ONS daemon propagates Oracle events. Refer to the section in this paper on *ONS Configuration* for more information. The Grid Infrastructure may be local or remote,

Copyright © 2025, Oracle and/or its affiliates / Public

and the utility can be used whether your Oracle clients are using JDBC, OCI, or ODP.Net managed and unmanaged providers.

Installing FANwatcher

Download the ZIP file from the blog: **Monitoring FAN Events**

On a Linux/Unix machine: expand the ZIP file on a machine containing an Oracle client installation or on any node of a Grid Infrastructure cluster.

Edit the file `FANwatcher` and update the line that sets `ORA_CRS_HOME`:

```
$ mkdir FANwatch

$ cd FANwatch

$ unzip FANwatcher.zip

Archive:  FANwatcher.zip

  inflating: ./classes/evtPrint.class

  inflating: ./src/evtPrint.java

  inflating: FANwatcher

  inflating: ./classes/fanWatcher.class

  inflating: ./src/fanWatcher.java

  inflating: ./classes/printEvtHeader.class

  inflating: ./src/printEvtHeader.java

  inflating: README.txt


$ vi FANwatcher

       ORA_CRS_HOME=/u01/app/oracle/product/12.1.0.2/GridInfrastructure
```

The directory path described by the ORA_CRS_HOME variable is specified as ORACLE_HOME when installing an Oracle client, Grid Infrastructure, or Oracle WebLogic server. The `FANwatcher` utility is looking for the pathname `$ORACLE_HOME/opmn/conf/ons.config` used by the ONS daemon to start.

The `FANwatcher` utility must be run by a user who has the privilege to access the Oracle home specified above. This privilege correlates to the user who installed the Oracle client, Grid Infrastructure (for example, `oracle`), or a member of the group to which this user belongs (for example, `oinstall`). If the user does not have the necessary privileges, an error similar to the following will occur when an attempt is made to run the utility:

Copyright © 2025, Oracle and/or its affiliates / Public

```
$ ./FANwatcher
>> Executing fanWatcher ...
Opening FAN Subscriber Window ...


15/01/27-04:05:47 ONS: config file (ons.config) could not be found.
Exception in thread "main" oracle.ons.ONSException: Unable to open config file
        at oracle.ons.ONS.readStandaloneLocalPort(ONS.java:1907)
        at oracle.ons.ONS.localInit(ONS.java:279)
        at oracle.ons.ONS.<init>(ONS.java:196)
        at oracle.ons.ONS.getONS(ONS.java:1156)
        at oracle.ons.Subscriber.<init>(Subscriber.java:162)
        at onc_subscriber.<init>(fanWatcher.java:34)
        at onc_subscriber.main(fanWatcher.java:117)
```

Using the FANwatcher utility

The FANwatcher utility takes optional arguments:

```
FANwatcher  <optional Notification type>
```

where.

The notification type defaults to all Oracle-generated FAN events (`database/event`). A subset could be specified, for example, `database/event/service` to see only service-related events, or `database/event/host` to see only node-related events.

To subscribe to all events (the most helpful case), use the following command:

```
FANwatcher
```

Events would then be received as shown by the following example:

Client Node	GI Node

`$ ./FANwatcher`
`>> Executing fanWatcher ...`
`Opening FAN Subscriber Window ...`	

	`$ srvctl start service –d testy –s testy_pdb_srv`

Copyright © 2025, Oracle and/or its affiliates / Public

```
** Event Header **
Notification Type: database/event/service
Delivery Time: Tue Jan 27 16:22:39 PST 2025
Generating Node: rac2.oracle.com
Event payload:
VERSION=1.0 event_type=SERVICEMEMBER service=testy_pdb_srv.us.oracle.com
instance=TESTY_3 database=testy db_domain=us.oracle.com host=rachost_725
status=up card=1 reason=USER timestamp=2025-01-27 16:22:39 timezone=-08:00

** Event Header **
Notification Type: database/event/service
Delivery Time: Tue Jan 27 16:22:39 PST 2025
Generating Node: rac2.oracle.com
Event payload:
VERSION=1.0 event_type=SERVICE service=testy_pdb_srv.us.oracle.com
database=testy db_domain=us.oracle.com host=rachost_725 status=up reason=USER
timestamp=2025-01-27 16:22:39 timezone=-08:00

** Event Header **
Notification Type: database/event/service
Delivery Time: Tue Jan 27 16:22:40 PST 2025
Generating Node: rac2.oracle.com
Event payload:
VERSION=1.0 event_type=SERVICEMEMBER service=testy_pdb_srv.us.oracle.com
instance=TESTY_1 database=testy db_domain=us.oracle.com host=rachost_723
status=up card=2 reason=USER timestamp=2025-01-27 16:22:40 timezone=-08:00

** Event Header **
Notification Type: database/event/service
Delivery Time: Tue Jan 27 16:22:41 PST 2025
Generating Node: rac2.oracle.com
Event payload:
VERSION=1.0 event_type=SERVICEMEMBER service=testy_pdb_srv.us.oracle.com
instance=TESTY_2 database=testy db_domain=us.oracle.com host=rachost_724
status=up card=3 reason=USER timestamp=2025-01-27 16:22:40 timezone=-08:00
```


Copyright © 2025, Oracle and/or its affiliates / Public

Server-side Fast Application Notification Callouts

FAN callouts provide a simple yet powerful integration mechanism available with RAC that can be deployed with minimal programmatic efforts. A FAN callout is a wrapper shell script or pre-compiled executable written in any programming language that is executed each time a FAN event occurs. The purpose of the FAN callout is to log in, file tickets, and take external actions. The purpose of the callout is not for integrated client failover – The FAN client failover is Fast Connection Failover in the next section. Except for node and network events, a FAN callout executes for the FAN events generated locally to each node and thus only for actions affecting resources on that node.

Configuring the FAN callout directory

Each FAN event posted by Grid Infrastructure results in the execution of each callout deployed in the standard Grid Infrastructure callout directory (on Linux, it is `${GI_Home}/racg/usrco`).

The order in which these callouts are executed is non-deterministic, and Grid Infrastructure guarantees that all callouts are invoked once for each FAN event in an asynchronous fashion. You can install as many callout scripts or programs as your system requires. FAN callouts, whose executions must be performed in a particular order, must be invoked from the same callout. Carefully test each callout for execution performance and correctness before production deployment.

Note: Ensure that the callout directory has write permissions only for the system user who installed Grid Infrastructure and that each callout executable or script contained therein has execute permissions only for the same Grid Infrastructure owner.

Implementing FAN callouts

Writing FAN callouts involves the following steps:

1. Parsing FAN payload
2. Filtering incoming FAN events
3. Executing event-handling programs

Parsing callout argument list

The first step in a FAN callout is parsing the FAN payload. As described in the FAN event taxonomy section, each FAN event type has a set of payload arguments.

The event type is displayed as the first argument in the FAN payload. For example:

```
SERVICEMEMBER VERSION=1.0 service=testy_pdb_srv.us.oracle.com database=testy instance=TESTY_1
host=rachost_723 status=down reason=USER timestamp=2025-01-27 17:56:08 timezone=-08:00
db_domain=us.oracle.com
```

A BASH shell script to parse the event could be as follows:

Copyright © 2025, Oracle and/or its affiliates / Public

```
#!/bin/bash
# Scan and parse HA event payload arguments:
#
NOTIFY_EVENTTYPE=$1 # Event type is handled differently
for ARGS in $*; do
PROPERTY=`echo $ARGS | $AWK -F"=" '{print $1}'`
VALUE=`echo $ARGS | $AWK -F"=" '{print $2}'`
case $PROPERTY in
VERSION|version) NOTIFY_VERSION=$VALUE ;;
SERVICE|service) NOTIFY_SERVICE=$VALUE ;;
DATABASE|database) NOTIFY_DATABASE=$VALUE ;;
INSTANCE|instance) NOTIFY_INSTANCE=$VALUE ;;
HOST|host) NOTIFY_HOST=$VALUE ;;
STATUS|status) NOTIFY_STATUS=$VALUE ;;
REASON|reason) NOTIFY_REASON=$VALUE ;;
CARD|card) NOTIFY_CARDINALITY=$VALUE ;;
VIP_IPS|vip_ips) NOTIFY_VIPS=$VALUE ;; #VIP_IPS for public_nw_down
TIMESTAMP|timestamp) NOTIFY_LOGDATE=$VALUE ;; # catch event date
TIMEZONE|timezone) NOTIFY_TZONE=$VALUE ;;
??:??:??) NOTIFY_LOGTIME=$PROPERTY ;; # catch event time (hh24:mi:ss)
esac
done
```

Filtering incoming events

You can filter the incoming events based on payload information. For example, you may decide to apply the following filtering criteria for each target event handler to be invoked from the callout:

Table 7: Example filtering criteria for specific FAN event handlers

Example Target Event Handler	Target Services to Filter	Event Type (and Status) to filter
Start or stop local applications	All	SERVICE (up or down) SERVICEMEMBER (up or down) NODE (node down)
Log a service request in help system	FIN_APAC, PROD	SERVICE (down) DATABASE (down) NODE (down)
Email or message IT Supervisor	All	NODE (nodedown) SERVICE (up or down)
Forward SNMP traps to remote management console	WEB_GL	all
Notify local vendor components	Custom remove applications	NODE (down)

Copyright © 2025, Oracle and/or its affiliates / Public

In the following BASH shell script example, a trouble ticket system (using the second filtering example in Table 5) is invoked only when the Oracle Grid Infrastructure framework posts a SERVICE, DATABASE, or NODE event type, with status either "down," "nodedown," or "public_nw_down" and only for two application service names: HQPROD and FIN_APAC:

```
#!/bin/bash
# FAN events with the following conditions will be inserted
# into the critical trouble ticket system:
# NOTIFY_EVENTTYPE => SERVICE | DATABASE | NODE
# NOTIFY_STATUS => down | public_nw_down | nodedown
# NOTIFY_DATABASE => HQPROD | FIN_APAC
#
if ((( [ $NOTIFY_EVENTTYPE = "SERVICE" ] ||
[ $NOTIFY_EVENTTYPE = "DATABASE" ] || \
[ $NOTIFY_EVENTTYPE = "NODE" ] \
) && \
( [ $NOTIFY_STATUS = "down" ] || \
[ $NOTIFY_STATUS = "public_nw_down" ] || \
 [ $NOTIFY_STATUS = "nodedown " ] \
)) && \
( [ $NOTIFY_DATABASE = "HQPROD" ] || \
[ $NOTIFY_DATABASE = "FIN_APAC" ] \
))
then
<< CALL TROUBLE TICKET LOGGING PROGRAM AND PASS RELEVANT NOTIFY_* ARGUMENTS >>
fi
```

This is an example of what can be done with callouts. Sample code to perform many tasks is available through OTN and Oracle Support: relocate services on a preferred node when an instance starts, remove client connections immediately when a service stops, and so forth.

An example of a FAN callout that would log all events generated on a given node could be constructed by:

1. Create a file in the callout directory:
   - `vi $ORA_CRS_HOME/racg/usrco/log_callouts.sh`
2. Add the following lines to the file (creating a BASH script):
   - ```
     #!/bin/bash
     echo "`date` : $@" >> [your_path]/admin/`hostname`/callout_log.log
     ```
3. Set the execute permission on the file
   - `chmod +x $ORA_CRS_HOME/racg/usrco/log_callouts.sh`
4. Copy this file to all nodes in the GI cluster and aggregate `callout_log.log` files to see all events on the system.

A sample `callout_log.log` entry may look like:

Copyright © 2025, Oracle and/or its affiliates / Public
     ```

```
Tue Jan 31 17:56:08 PST 2025: SERVICEMEMBER VERSION=1.0
service=testy_pdb_srv.us.oracle.com database=testy instance=TESTY_1
host=rachost_723 status=down reason=USER timestamp=2025-01-27 17:56:08 timezone=-
08:00 db_domain=us.oracle.com
```

More information on FAN and FAN Callouts can be found in Chapter 5, Workload Management with Dynamic Database Services of the *Oracle® Real Application Clusters Administration and Deployment Guide 23ai,* Part Number F46762-03

Copyright © 2025, Oracle and/or its affiliates / Public

Configuring Applications to Use FAN with Fast Connection Failover

Fast Connection Failover (FCF) is a pre-configured and proven FAN client-side integration. The application code does not need to be modified to receive and use FAN events. You should always use an FCF-capable client.

FCF is the FAN client solution for all application stacks, whether Oracle or third-party. An FCF client automatically subscribes to and acts on FAN events. FCF clients can understand which database sessions are impacted by an event by using the FAN connection signature shown in Table 1. By matching the payload in the FAN event with that in the connection signatures, FCF clients know which sessions to act on when a FAN event is received. FCF operates as follows based on the event type (Table 2), event status (Table 3), and reason (Table 4):

- Down - Events with `status=down` and `reason=FAILURE` are caused by failures at the database server. The FCF client immediately aborts the related connections so that the application does not hang on TCP/IP timeouts but receives an interrupt quickly. The dead connections are removed from pooled FCF clients. When the application is configured to use Application Continuity or TAF, the application automatically fails over to the other instance.
- Planned Down - Events with `status=down` and `reason=PLANNED` are posted when the DBA stops an instance, service, or database to start planned maintenance using `srvctl, gdsctl,` or `Data Guard Broker`. The FCF client drains the sessions ahead of the planned maintenance by allowing the active work to be completed and then closing the session. There is no application interruption for planned draining by FAN and FCF. For Oracle Database 12c or later UCP and JDBC clients, the system property `oracle.ucp.PlannedDrainingPeriod` allows for a period to be set over which the draining occurs.
- Up - Events with `status=up` are posted when a service starts for the first time or resumes. The FCF client reallocates the sessions to balance the load across the database server. Rebalancing is a gradual process that does not impact performance.
- Load% - When runtime load balancing is enabled, the FAN events carry an advised percentage to distribute load across the service. The FCF client uses this advice to balance sessions locally when using RAC and globally when using GDS.
- Affinity - When runtime load balancing is enabled, the FAN events advise when to keep the client conversation local. This advice applies to repeated borrows and returns from the same client.
- Load % and Affinity are applicable to pooled FCF clients (see Table 8). All other actions apply to all FCF clients

FCF is provided with the following clients, beginning with the releases listed in Table 8. Table 8 also shows the transport over which FAN is received by release.

Copyright © 2025, Oracle and/or its affiliates / Public

Table 8. Fast Connection Failover by Oracle Database Release

FCF Client	Database Version 11g	12c / 19c /23ai
JDBC Implicit Connection Cache (ICC)	ONS	ICC deprecated
JDBC Universal Connection Pool (UCP)	ONS	ONS
WebLogic Server Active GridLink	ONS	ONS
3rd Party Application Servers using UCP: Apache TomCat, IBM WebSphere	ONS	ONS
ODP.Net Unmanaged (OCI)	AQ	ONS
ODP.Net Managed (C#)	ONS	ONS
OCI Session Pool	AQ	ONS
PHP	AQ	ONS
SQL*Plus	AQ	ONS
Tuxedo	ONS	ONS
JDBC Thin Standalone Client	ONS	ONS
OCI/OCCI Driver	AQ	ONS
Net and SCAN Listeners	ONS	ONS

Beginning with Oracle Database 12c Release 1 (12.1), ONS is the transport when using an Oracle Database. The FAN AQ HA notification feature is deprecated and maintained only for backward compatibility when an old component (Oracle Database 11g) is involved.

How to Configure the Oracle Notification Service for 23ai Grid Infrastructure

FAN uses the Oracle Notification Service (ONS) for event propagation to all clients from Oracle Database 12c onwards. ONS is installed as part of Oracle Grid Infrastructure on a cluster, in an Oracle Data Guard installation, and when Oracle WebLogic is installed. ONS propagates FAN events to all other ONS daemons it is registered with. ONS comes pre-configured and enabled with Oracle Grid Infrastructure. It is not necessary to manually configure ONS for it to operate.

Grid Infrastructure comes pre-configured with FAN. No steps are needed to configure or enable FAN at the server-side with one small exception: OCI FAN and ODP FAN require `–notification` is set to `TRUE` for the service by `srvctl` or `gdsctl`. For variations on the server side, refer to the appendix.

For FCF clients, ONS is installed as part of the WebLogic Server and as part of the Oracle client installation. With FAN auto-configuration at the client, ONS must be on the CLASSPATH or in the ORACLE_HOME, depending on your client. Transparently, the FCF client spawns an ONS thread to subscribe to interested FAN events.

Refer to *Appendix A: Configuring ONS* in this paper for more information on ONS configuration.

Copyright © 2025, Oracle and/or its affiliates / Public

General Steps for Configuring FCF Clients

Follow these steps before progressing to driver-specific instructions:

- Use a dynamic database service
- Use the Oracle notification service
- How many ONS connections are needed?
- How to pass ONS through Firewalls
- Use a NET connection that provides high availability
- Enable connection checking

Use a Dynamic Database Service

Using FAN requires that the application connect to the database using a dynamic database service. This service is created using `srvctl if RAC or gdsctl` if global database services is used. Do not connect using the default database or PDB service—these are for administration only and are not supported for FAN. The TNSnames entry or URL must use the service name syntax and follow best practices by specifying a dynamic database service name. Refer to the examples later in this section.

Use the Oracle Notification Service

When using FAN with JDBC thin, Tuxedo, Oracle Database OCI, or ODP.Net clients, FAN is received over ONS. When you use Oracle Database 11g OCI or ODP.NET unmanaged provider FCF clients, FAN is received over AQ.

The target for FAN clients of Oracle Database 12*c* and higher versions is no configuration. ONS FAN auto-configuration ensures that FCF clients discover the server-side ONS networks and self-configure. FAN is automatically enabled when ONS libraries or jars are present. In Oracle Database 11g, enabling FAN on most FCF clients is still necessary. Listeners and SQL*Plus require no client-side configuration.

FAN auto-configuration removes the need to list the Grid Infrastructure servers an FCF client needs. Clients already use a TNS address string or URL to locate the remote listeners. FAN auto-configuration uses the TNS addresses to locate the databases and then asks each server database for the ONS server-side addresses. When there is more than one database, for example, FAN auto-configuration contacts each and obtains an ONS configuration for each one.

When using Oracle Database 23ai, the ONS network is discovered from the URL. An ONS node group is automatically obtained for each address list when `LOAD_BALANCE` is off across the address lists. This is the default shown here. If `LOAD_BALANCE` is `TRUE` or `ON` across all address lists, only one set of ONS endpoints is created. (Note that this is the `LOAD_BALANCE` value across the `ADDRESS_LIST`. The `LOAD_BALANCE` inside each address list is to expand the SCAN address.)

How many ONS connections are needed?

By default, the FCF client maintains three hosts for redundancy in each node group in the ONS configuration. Each node group corresponds to each Grid Infrastructure cluster or each GDS data center. For example, if there is a primary database and several Oracle Data Guard standbys, there are, by default, 3 ONS connections maintained at each node group. The node groups are discovered using FAN auto-configuration (Oracle Database 12*c* uses `ons configuration`). With node_groups defined by FAN auto-configuration, `load_balance=false` (the default), more ONS endpoints are not required. Increase the max connections to increase the number of endpoints. This applies to each node group. Increasing to 4 in this example maintains four ONS connections at each node. Increasing this value consumes more sockets.

```
oracle.ons.maxconnections=4
```

ONS Node groups: `oracle.ons.nodes`

If the client is to connect to multiple clusters and receive FAN events from both, for example, in the RAC with a Data Guard situation, then multiple ONS node groups are needed. FAN auto-configuration creates these node

Copyright © 2025, Oracle and/or its affiliates / Public

groups using the URL or TNS names for 12*c* or later client and database. If auto-ons is not used, specify the node groups in the `oraaccess.xml` configuration file.

Consider the situation with two clusters of 8 nodes each. Specify two node lists (one for each cluster):

```
AUTOONS creates a separate node group for each addresslist in the TNS connection
descriptor:

oracle.ons.nodes.001=node1a:6250,node1b:6250,node1c:6250,node1d:6250,node1e:6250,n
ode1f:6250,node1g:6250,node1h:6250

oracle.ons.nodes.002=node2a:6250,node2b:6250,node2c:6250,node2d:6250,node2e:6250,n
ode2f:6250,node2g:6250,node2h:6250
```

Leaving `oracle.ons.maxconnections` at the default of 3, for every active node list, results in the ONS client trying to maintain 6 total connections in this case.

How to pass ONS through Firewalls

When a firewall is present between the ONS node groups and the FCF client, a port needs to be opened for ONS traffic. The recommended approaches are –

1. Add the `ons executable` to the firewall exceptions list.
2. Add the port used by ONS to the firewall's exceptions list.

Use a NET Connection that provides High Availability

For OCI and ODP.NET unmanaged provider clients, use the following TNS names structure:

```
Alias = (DESCRIPTION = (CONNECT_TIMEOUT=90)(RETRY_COUNT=100)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=1000ms)

(ADDRESS_LIST =

  (LOAD_BALANCE=on)

  (ADDRESS = (PROTOCOL = TCP)(HOST=primary-SCAN)(PORT=1521)))

(ADDRESS_LIST =

  (LOAD_BALANCE=on)

  (ADDRESS = (PROTOCOL = TCP)(HOST=secondary-SCAN)(PORT=1521)))

(CONNECT_DATA=(SERVICE_NAME = gold-cloud)))

For older (11g, 12.1.0.2) JDBC thin clients, use the following URL structure:

jdbc:oracle:thin = (CONNECT_TIMEOUT=4) (RETRY_COUNT=30)(RETRY_DELAY=3)

(ADDRESS_LIST =

  (LOAD_BALANCE=on)

  ( ADDRESS = (PROTOCOL = TCP)(HOST=primary-SCAN)(PORT=1521)))

(ADDRESS_LIST =

  (LOAD_BALANCE=on)

  ( ADDRESS = (PROTOCOL = TCP)(HOST=secondary-SCAN)(PORT=1521)))

(CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

Note that after Oracle Database 12*c* Release 1 (12.1.0.2), JDBC and OCI align, and you should use the OCI version for all connection descriptions.

Copyright © 2025, Oracle and/or its affiliates / Public

Best Practices for the NET Connections

ALWAYS use dynamic database services to connect to the database. These are created using `srvctl` or `gdsctl`.

Do not use the default database or PDB service - these are for administration only, not for application usage, and do not provide FAN and many other features because they are available at mount.

For JDBC, use the latest client driver (Oracle Database 23ai) with current or older RDBMS.

Use one `DESCRIPTION` in the TNS names entry or URL - using more causes long delays connecting when `RETRY_COUNT` and `RETRY_DELAY` are used.

Set `CONNECT_TIMEOUT=90` or higher to prevent logon storms for OCI and ODP clients. Use a lower setting for 11g JDBC clients - `CONNECT_TIMEOUT=4,` as a temporary measure, as `TRANSPORT_CONNECT_TIMEOUT` is unavailable.

Do not also set the JDBC property `oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR` as it overrides `CONNECT_TIMEOUT.`

Set `LOAD_BALANCE=ON` per address to expand SCAN names starting with Oracle Database 11g Release 2 (11.2.0.3) for OCI and Oracle Database 12*c* Release 1 (12.1.0.2) for JDBC.

Enable Connection Checks

Depending on the outage, applications may receive stale connections when connections are borrowed before FCF is processed. This can occur, for example, on a clean instance down when sockets are closed coincident with incoming connection requests. To prevent the application from receiving any errors, connection checks should be enabled at the connection pool.

JDBC Universal Connection Pool

setValidateConnectionOnBorrow(boolean) - Specifies whether or not connections are validated when the connection is borrowed from the connection pool. The method enables validation for every connection that is borrowed from the pool. The default value is false. Set the value to true so validation is performed.

OCI Connections

To verify that either a FAN or an OCI_ERROR terminates the connection to the server, an application can check the value of the attribute OCI_ATTR_SERVER_STATUS in the server handle. If the attribute's value is OCI_SERVER_NOT_CONNECTED, then the connection to the server has been terminated, and the user session should be reestablished.

OCI_ATTR_SERVER_STATUS is set to OCI_SERVER_NOT_CONNECTED for both FAN unplanned down and FAN planned down use cases. When using FAN to report unplanned downtime, the application receives an error immediately. When using FAN to report planned maintenance, a custom OCI pool can check OCI_ATTR_SERVER_STATUS before borrowing and after returning to the pool and dropping the session at these safe places. Dropping closed connections before borrowing and after returning leads to a good user experience with no application errors received during planned maintenance.

ODP.NET Provider

CheckConStatus is on by default.

This property checks the status of the connection before putting the connection back into the ODP.NET connection pool. The installation of ODP.NET does not create this registry entry. However, the default value 1 is used.

For custom ODP.NET programs, test ConnectionState.Open. For example

Copyright © 2025, Oracle and/or its affiliates / Public

```
if(OracleConnection.State!=ConnectionState.Open)

  OracleConnection.Open()
```

How to Configure FAN for 23ai Java Clients

Using Universal Connection Pool

The best way to take advantage of FCF with the Oracle Database JDBC thin driver is to use the Universal Connection Pool (UCP) or WebLogic Server Active GridLink. Setting the pool property `FastConnectionFailoverEnabled` on the Universal Connection Pool enables Fast Connection Failover (FCF). Active GridLink always has FCF enabled by default. Third-party application servers, including IBM WebSphere and Apache Tomcat, support UCP as a connection pool replacement. For more information on embedding UCP with other web servers, refer to these technical briefs:

*Design and Deploy WebSphere Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP (*http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-websphere-2409214.pdf) and

*Design and deploy Tomcat Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP (*http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-tomcat-2265175.pdf).

Follow these configuration steps to enable Fast Connection Failover:

1. The connection URL of a connection factory must use the service name syntax and follow best practices by specifying a dynamic database service name and the JDBC URL structure (above and also below). All other URL formats do not provide high availability. The URL may use JDBC thin or JDBC OCI.
2. If wallet authentication has not been established or the cluster is running a version earlier than Oracle Grid Infrastructure 12.1.0.2, remote ONS configuration is needed. This can be done through the pool property `setONSConfiguration`, which can be set through a property file, as shown in the following example:

```
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();

pds.setConnectionPoolName("FCFSamplePool");

pds.setFastConnectionFailoverEnabled(true);

pds.setONSConfiguration("propertiesfile=/usr/ons/ons.properties");

pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");

pds.setURL("jdbc:oracle:thin@((CONNECT_TIMEOUT=90)(RETRY_COUNT=100)"+
" (RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=1000ms)"+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  " ( ADDRESS = (PROTOCOL = TCP)(HOST=RAC-SCAN)(PORT=1521))) "+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  "( ADDRESS = (PROTOCOL = TCP)(HOST=DG-SCAN)(PORT=1521)))"+

"(CONNECT_DATA=(SERVICE_NAME=service_name)))");
```

The property file specified must contain an `oracle.ons.nodes` property and optionally, properties for `oracle.ons.walletfile` and `oracle.ons.walletpassword`. An example of an `ons.properties` file is shown here:

```
oracle.ons.nodes=racnode1:4200,racnode2:4200
```


Copyright © 2025, Oracle and/or its affiliates / Public

3. Ensure the pool property `setFastConnectionFailoverEnabled=true` is set.
4. The `CLASSPATH` must contain `ons.jar, ucp.jar`, and the jdbc driver jar file, for example `ojdbc11.jar`.
5. The following is not required and is an optimization to force re-ordering of the SCAN IP addresses returned from DNS for a SCAN address:

```
oracle.jdbc.thinForceDNSLoadBalancing=true
```

6. If you are using JDBC thin with Oracle Database 23ai, Application Continuity can be configured to failover the connections after FAN is received.
7. If the database is earlier than 12c or if the configuration needs different ONS endpoints from those auto-configured, the ONS endpoints can be enabled as per the following example:

```
pds.setONSConfiguration("nodes=mysun05:6200,mysun06:6200,
mysun07:6200,mysun08:6200");
```

Or in the situation with multiple clusters and when using autoons. autoons would generate node lists similar to the following:

```
oracle.ons.nodes.001=node1a:6250,node1b:6250,node1c:6250,node1d:6250,node1e:6250,n
ode1f:6250,node1g:6250,node1h:6250

oracle.ons.nodes.002=node2a:6250,node2b:6250,node2c:6250,node2d:6250,node2e:6250,n
ode2f:6250,node2g:6250,node2h:6250
```

`oracle.ons.maxconnections` is set to 3 by default for EVERY active nodelist, so there is no need to set this explicitly. This example will result in the ONS client trying to maintain 6 total connections.

Copyright © 2025, Oracle and/or its affiliates / Public

How to Configure FAN for ODP.Net Clients, Managed and Unmanaged Providers

The application should use the ODP.Net Connection Pool to take advantage of Fast Connection Failover (FCF) with an Oracle Database unmanaged or ODP.NET managed provider. FCF is supported for the connection pools and Oracle Services for Microsoft Transaction Server (OraMTS).

Configuration for Oracle Database 23ai ODP.Net Unmanaged and Managed Provider Clients with 23ai Database Server

1. To enable FAN with ODP.NET on the client side, specify HA events in the connect string:

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;“
```

2. To enable Runtime Load balancing with ODP.NET at the client side, also specify load balancing in the connect string:

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;
load balanacing=true“
```

3. ODP.Net needs to be able to find the ONS listeners. Using FAN autoons, ONS configuration information is passed directly to ODP.Net from the database server. Autoons uses the TNSnames to find the endpoints. Set the OCI TNS address best practice with full high availability capabilities.

```
myAlias=(DESCRIPTION= (CONNECT_TIMEOUT=90)(RETRY_COUNT=100)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=1000ms)

(ADDRESS_LIST=(LOAD_BALANCE=ON)

  (ADDRESS=(PROTOCOL=TCP)(HOST=primary-scan)(PORT=1521)))

(ADDRESS_LIST=(LOAD_BALANCE=ON)

  (ADDRESS=(PROTOCOL=TCP)(HOST=standby-scan)(PORT=1521)))

 (CONNECT_DATA=(SERVICE_NAME=service_name)))
```

Transparent Application Failover (TAF) can be used with FCF and ODP.Net unmanaged providers. After FAN interrupts the session, TAF executes the failover using the `FAILOVER_TYPE` specified – `BASIC` or `SELECT`. Application Continuity will be available in a future release.

Which Steps Differ if Using ODP.Net UnManaged Provider 11g or ODP.Net UnManaged Provider with Oracle Database Release 11g?

When using an earlier database or client version than Oracle Database 12c with an unmanaged provider, the steps are the same as those for 12c client and 12c database, with one exception. FAN events are delivered over AQ, which requires `notification` to be set on the database service, similar to the following example:

```
srvctl modify service -db EMEA -service GOLD -notification TRUE
```


Copyright © 2025, Oracle and/or its affiliates / Public

Which steps differ if you want to custom configure ONS endpoints with 23ai?

For a customized ONS configuration

Edit oraaccess.xml located in $ORACLE_HOME/network/admin.

```
<onsConfig mode=”remote”>
   <ons database=”db1”>
      <add name=”nodeList” value=”racnode1:6700, racnode2:6700” />
   </ons>
   <ons database=”db2”>
      <add name=”nodeList” value=”racnode3:6700, racnode3:6700” />
   </ons>
</onsConfig>
```

In the case of custom ONS configuration, the application specifies the <host>:<port> values for every potential database it can connect to. The <host>:<port> value pairs represent the ports on the different Oracle RAC nodes where the ONS daemons talk to their remote clients.

Additional information on configuring ONS is available in the Appendix attached to this paper.

Refer to the following for FAN and ODP.Net Provider Oracle® Data Provider for .NET Developer's Guide 23ai Part Number F46984-09.

Copyright © 2025, Oracle and/or its affiliates / Public

How to Configure FAN for OCI Clients

OCI clients embed FAN at the driver level so all clients can use it regardless of the pooling solution. Starting with Oracle database and client 12*c*, OCI clients use ONS for the FAN transport. Both the server and the client must use version 12*c* or later. If either the client or server uses version 11.2 or earlier, the FAN transport used is advanced queues.

Configuration for SQL*Plus and PHP

1. Set notification for the service

```
srvctl modify service -db EMEA -service GOLD -notification TRUE
```

2. For PHP clients only, add oci8.events=On to php.ini:

```
php.ini:

oci8.events=on
```

Important

If oraccess.xml is present with events=-false or events not specified, this disables the usage of FAN. To maintain FAN with SQL*Plus and PHP when oraccess.xml is in use, set events=-true

3. When using the 23ai client with the Oracle Database 23ai database, enable FAN in `oraaccess.xml` on the client side.

```
<oraaccess> xmlns="http://xmlns.oracle.com/oci/oraaccess"
  xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
  schemaLocation="http://xmlns.oracle.com/oci/oraaccess
  http://xmlns.oracle.com/oci/oraaccess.xsd">
  <default_parameters>
    <events>true</events>
  </default_parameters>
</oraaccess>
```

Configuration for 23ai OCI Clients with Oracle 23ai Database

1. Tell OCI where to find ONS Listeners

Starting with 12*c*, the client install comes with ONS linked into the client library. Using ONS auto-config, the ONS endpoints are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using `oraaccess.xml`.

2. Enable FAN high availability events for the OCI connections

To enable FAN requires editing the OCI file `oraaccess.xml` to specify the global parameter `events`. This file is located in `$ORACLE_HOME/network/admin`:

Copyright © 2025, Oracle and/or its affiliates / Public

```
<oraaccess> xmlns="http://xmlns.oracle.com/oci/oraaccess"
  xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
  schemaLocation="http://xmlns.oracle.com/oci/oraaccess
  http://xmlns.oracle.com/oci/oraaccess.xsd">
  <default_parameters>
    <events>true</events>
  </default_parameters>
</oraaccess>
```

3. Tell OCI where to find ONS Listeners.

   Starting with 12*c*, the client install comes with ONS linked into the client library. Using ONS auto-config, the ONS endpoints are discovered from the TNS address. This automatic method is the recommended approach. Like ODP.Net, manual ONS configuration is also supported using `oraaccess.xml`.

```
myAlias=(DESCRIPTION= (CONNECT_TIMEOUT=90)(RETRY_COUNT=100)(RETRY_DELAY=3)
(TRANSPORT_CONNECT_TIMEOUT=1000ms)
(ADDRESS_LIST=(LOAD_BALANCE=ON)
  (ADDRESS=(PROTOCOL=TCP)(HOST=primary-scan)(PORT=1521)))
(ADDRESS_LIST=(LOAD_BALANCE=ON)
  (ADDRESS=(PROTOCOL=TCP)(HOST=standby-scan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME=service_name)))
```

4. Enable FAN at the server for all OCI clients.

   Enabling FAN on the Oracle 23ai database server for all OCI clients (including SQL*Plus) is still necessary.

```
srvctl modify service -db EMEA -service GOLD -notification TRUE
```

Which steps differ if you want to custom configure ONS endpoints with Oracle Database 23ai

If you want to custom configure ONS, use syntax similar to the following example. This is not the recommended method.

Copyright © 2025, Oracle and/or its affiliates / Public

```
<oraaccess xmlns="http://xmlns.oracle.com/oci/oraaccess"
  xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
  schemaLocation="http://xmlns.oracle.com/oci/oraaccess
  http://xmlns.oracle.com/oci/oraaccess.xsd">
  <default_parameters>
    <fan>
     <!-- only possible values are "trace" or "error"  -->
      <subscription_failure_action>
        error
      </subscription_failure_action>
    </fan>
    <ons>
      <subscription_wait_timeout>
        5
      </subscription_wait_timeout>
      <auto_config>true</auto_config>
      <!—The following provides the manual configuration for ONS.
          Note that this functionality should not need to be used
          as auto_config can normally discover this information. -->
      <servers>
        <address_list>
          <name>pacific</name>
          <max_connections>3</max_connections>
          <hosts>10.228.215.121:25293, 10.228.215.122:25293</hosts>
        </address_list>
        <address_list>
          <name>Europe</name>
          <max_connections>3</max_connections>
          <hosts>myhost1.mydomain.com:25273,
            myhost2.mydomain.com:25298,
            myhost3.mydomain.com:30004</hosts>
        </address_list>
      </servers>
    </ons>
    <events>true</events>
  </default_parameters>
</oraaccess>
```

Copyright © 2025, Oracle and/or its affiliates / Public

OCI and Oracle Database 11g

If the client or the database uses Oracle Database 11g or earlier, FAN events are sent over Advanced Queues (AQ).

For an Oracle Database 11g application, the application itself must:

- Initialize the OCI Environment in OCI_EVENTS mode
- Link with a thread library
- Set `–notification` for the service using `srvctl` or `gdsctl`

```
srvctl modify service -db EMEA -service GOLD -notification TRUE
```

For more information, refer to *Oracle® Call Interface Developer's Guide 23ai* F46705-09*, Chapter 11, High Availability in OCI*.

Copyright © 2025, Oracle and/or its affiliates / Public

Conclusion

FAN - Tells the applications when actions are required

To achieve fast recovery and mitigate flow-on effects after a failure of the hardware or the software supporting the application session, the session must be interrupted immediately when a failure is detected.

For service configuration changes, Grid Infrastructure and GDS post FAN events immediately when a state change occurs for services in the system. Instead of waiting for the application server or driver to time out and detect a problem, the application server or driver receives these events and acts immediately using FAN.

For unplanned down events, the disruption to the application is minimized as connections to the failed instance or node are terminated, sockets are closed, in-flight requests are terminated, and the application user is notified immediately. If using TAF for OCI or Application Continuity for Java, the user experiences a slight disruption while the session is re-established at a functioning service. Not-borrowed sessions are cleaned up immediately, and application users requesting connections are directed to instances offering functioning services only.

For planned down events, the disruption to the application is minimized by dropping the session at a safe place when returned to the connection pool, where the application receives no error whatsoever. The same is true for unplanned, not-borrowed sessions, which are cleaned up immediately, and application users requesting connections are directed to instances offering functional services only.

When services are started, new connections are created automatically for Up events so the application can immediately take advantage of the extra resources.

The FANwatcher utility described in this paper enables you to determine the ONS topology that has been constructed and whether a subscribing client can receive an event.

Copyright © 2025, Oracle and/or its affiliates / Public

Appendix A: Configuring ONS

ONS is installed as part of Oracle Grid Infrastructure on a cluster, in an Oracle Data Guard installation, and when Oracle WebLogic Server Active Gridlink is installed. ONS propagates FAN events to all other ONS daemons it is registered with. There is no need to configure ONS manually. ONS will also be installed as part of an Oracle client installation and may be spawned by a standalone client application through a Remote ONS configuration process.

Oracle application clients will subscribe, through an ONS daemon, to events in which it, the client, is interested.

ONS Configuration File

The ONS configuration file, `$ORACLE_HOME/opmn/conf/ons.config`, defines how ONS behaves. Information in this file is stored as a set of key-value pairs, as shown in Table 9.

Table 9. ons.config Parameters and Definition

Parameter	Definition
`localport`	The port number on the local host that ONS will use to talk to local clients. Default in Grid Infrastructure is `localport=6100`
`remoteport`	The port number used by ONS to talk to other ONS daemons. The default value in Grid Infrastructure is `remoteport=6200`
`nodes`	A comma-separated list of nodes and ports indicating other ONS daemons to talk to. The port value is the remoteport entry that each ONS daemon will be listening on. In a Grid Infrastructure configuration all nodes in the cluster will be named. An example would look like nodes=myhost1.example.com:6500,myhost2.example.com:6500,
`logcomp` (optional parameter)	An optional parameter that specifies what subcomponents to log.The format is `<component>[<subcomponent>,…]` Exclusion of subcomponents is also allowed by preceding the subcomponent by an exclamation mark (!). For example, to log `all` components, except for `secure` specify `logcomp=ons[all,!secure]`
`logfile` (optional parameter)	The location of the log file used for logging messages. The default value is `logfile=$ORACLE_HOME/opmn/logs/ons.log`
`walletfile` (optional parameter)	Specifies the wallet file used by the Oracle Secure Sockets Layer (SSL) to store SSL certificates. If a wallet file is specified to ONS, then it uses SSL when communicating with other ONS instances and require SSL certificate authentication from all ONS instances that try to connect to it. In 12.1.0.2 a Grid Infrastructure installation will have the `walletfile` set by default, thus enforcing the use of SSL for all ONS connections. Existing UCP users will need to verify that they are using a walletfile, and ensure the contents are the same as the wallet on the server.
`useocr` (optional GI-SERVER ONLY parameter)	This parameter is only to be used on a Grid Infrastructure node. It indicates whether ONS should store all GI configuration in the Oracle Cluster Registry (OCR). To store information in OCR use `useocr=on`
`allowgroup` (optional parameter)	Specifies the ONS setting to indicate the user group connecting to the `localport`. When set to true, ONS allows users within the same OS group to connect to its local port. When set to `false`, ONS only allows the user who started the ONS daemon to access its local port. The default value of this parameter is `false`. When

Copyright © 2025, Oracle and/or its affiliates / Public

	using remote ONS configuration, there is no need to set this parameter.

Note that the modification of these parameters in a Grid Infrastructure installation is done using srvctl with the modify nodeapps command. The following help screen excerpt shows the relevant parameters:

```
$ srvctl modify nodeapps -h

Modifies the configuration for a node application.

Usage: srvctl modify nodeapps {[-node <node_name> -address
{<vip_name>|<ip>}/<netmask>[/if1[|if2...]] [-skip]] | [-subnet
<subnet>/<netmask>[/if1[|if2|...]]]} [-nettype {STATIC|DHCP|AUTOCONFIG|MIXED}]
[-emport <em_port>] [ -onslocalport <ons_local_port> ] [-onsremoteport
<ons_remote_port> ] [-remoteservers <host>[:<port>][,<host>[:<port>]...]]
[-clientdata <file>] [-pingtarget "<pingtarget_list>"] [-verbose]
  -node <node_name>              Node name
  -onslocalport  <ons_local_port> ONS listening port for local client
connections
  -onsremoteport <ons_remote_port> ONS listening port for connections from
remote hosts
  -remoteservers                 <host>[:<port>][,<host>[:<port>]...]  List
of remote host/port pairs for ONS daemons outside this cluster
  -clientdata <file>             file with wallet to import, or empty string
to delete wallet used for SSL to secure ONS communication
  -verbose                       Verbose output
  -help                          Print usage
```


Copyright © 2025, Oracle and/or its affiliates / Public

Client-side ONS Configuration

CLIENT-SIDE ONS IS NOT RECOMMENDED PRACTICE. For applications, use AUTO-ONS or remote ONS.

For Oracle clients that require an ONS daemon to be running, it is necessary to edit the `ons.config` file and restart the ONS daemon.

The default location for an Oracle Client installation is `$ORACLE_HOME/opmn/conf/ons.config`, although this may vary for other Oracle product installations.

A sample Oracle Client `ons.config` file could look like:

```
# This is an example ons.config file
#
# The first three values are required
localport=4100
remoteport=4200
nodes=racnode1.example.com:6200,racnode2.example.com:6200
```

As indicated in this example, it is necessary to specify the cluster nodes on which RAC instances will run, so that this client will receive FAN events for which it is interested. Specifying all cluster nodes is unnecessary, as ONS will discover daemons running within the topology it constructs. However, if only one or a subset of nodes is specified in the `nodes=...` list, there is a risk that this node may be down when the ONS topology is being discovered and constructed.

The `$ORACLE_HOME/opmn/bin/onsctl` utility can be used to `start` the ONS daemon.

The `onsctl` command

`onsctl` can be used to start, shutdown, reconfigure, and monitor the ONS daemon. Available command options to `onsctl` are:

Table 10. onsctl commands, action, and output

Command	Action	Output
`Start`	Start the ONS daemon	`onsctl: ons started`
`Shutdown`	Stop the ONS daemon	`onsctl: shutting down ons daemon`
`Reload`	Re-read the `ons.config` file and update settings (ONS daemon not stopped)	
`ping [max-retry]`	Verifies the local ONS daemon is running. Will attempt `max-retry` times	`ons is running`
`Debug`	Print debug information for the local ONS daemon.	
`Usage`	Print a detailed help screen.	
`Help`	Print simple usage information	

Note that ONS is managed as part of Grid Infrastructure if it is running in a cluster. The `srvctl` utility allows for ONS start and stop through the `nodeapps` option, as shown by this help screen:

Copyright © 2025, Oracle and/or its affiliates / Public

```
$ srvctl start nodeapps -h
Start the node applications running on a node.

Usage: srvctl start nodeapps [-node <node_name>] [-adminhelper | -onsonly]
[-verbose]
  -node <node_name>         Node name
  -adminhelper              Start Administrator helper only
  -onsonly                  Start ONS only
  -verbose                  Verbose output
  -help                     Print usage
```

Validating ONS Topology

The `onsctl debug` command produces output that can be useful in determining the topology being used by the ONS daemons. It shows the server:port combinations on which ONS daemons are present, thus capable of receiving FAN events.

An edited sample of the `onsctl debug` output from a 4-node Grid Infrastructure cluster is shown here:

Copyright © 2025, Oracle and/or its affiliates / Public

```
== rac1.oracle.com:6200 5844 15/01/28 17:50:50 ==


Listener:
  TYPE                BIND ADDRESS               PORT  SOCKET
-------- ------------------------------------- ----- ------
Local                                     ::1  6100      6
Local                               127.0.0.1  6100      7
Remote                                  any  6200      8
Remote                                  any  6200      -


Connection Topology: (4)
                IP                       PORT   VERS  TIME
------------------------------------ ----- ----- --------
                          10.10.10.247  6200     4 54c5a3e3
                           **                          10.10.10.244 6200
                           **                          10.10.10.245 6200
                           **                          10.10.10.246 6200
                          10.10.10.246  6200     4 54c5a3e3
                           **                          10.10.10.244 6200
                           **                          10.10.10.245 6200
                           **                          10.10.10.247 6200
                          10.10.10.245  6200     4 54c5a3e3
                           **                          10.10.10.244 6200
                           **                          10.10.10.247 6200
                           **                          10.10.10.246 6200
                          10.10.10.244  6200     4 54c5a3e3=
                           **                          10.10.10.247 6200
                           **                          10.10.10.245 6200
                           **                          10.10.10.246 6200


Server connections:
   ID            CONNECTION ADDRESS              PORT  FLAGS  SENDQ REF WSAQ
-------- ------------------------------------- ----- ------ ----- --- ----
       0                           10.10.10.245  6200 010405 00000 001
       1                           10.10.10.246  6200 010405 00000 001
       2                           10.10.10.247  6200 010405 00000 001
```


Copyright © 2025, Oracle and/or its affiliates / Public

The `Listener section` shows the `localport` and `remoteport` Port numbers being communicated over.

The section titled `Connection Topology: (4)` shows the IP addresses and port numbers where an ONS daemon is running (that the daemon that generated this output is aware of). In this case, the daemon running on **`rac1.oracle.com:6200`** is communicating with ONS daemons at the IP address:Port combinations: `10.10.10.247:6200, 10.10.10.246:6200, 10.10.10.245:6200,` and `10.10.10.244:6200` (which is the local daemon running on `rac1.oracle.com`). Daemons that each of these ONS processes is aware of are shown as subentries.

The section titled `Server connections:` shows all ONS daemons other than the local daemon that are visible.

Remote ONS Configuration

UCP for JDBC supports ONS configuration through the `ONSConfiguration` pool property. This property is used to set the remote ONS configuration. The parameter string closely resembles the content of the ONS configuration file (`ons.config`). The string contains a list of name=value pairs separated by a newline character (`\n`). The name can be one of `nodes, walletfile,` or `walletpassword`.

The parameter string should at least specify the ONS configuration nodes attribute as a list of host:port pairs separated by a comma. SSL would be used when the `walletfile` attribute is defined as an Oracle wallet file.

A JDBC code example of setting the ONS configuration string on a `PoolDataSource` is:

```
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();


pds.setConnectionPoolName("FCFSamplePool");

pds.setFastConnectionFailoverEnabled(true);

pds.setONSConfiguration("nodes=racnode1:4200,racnode2:4200\nwalletfile=

/oracle11/onswalletfile");

pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");

pds.setURL("jdbc:oracle:thin@((CONNECT_TIMEOUT=90)(RETRY_COUNT=100)"+
" (RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=1000ms)"+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  " ( ADDRESS = (PROTOCOL = TCP)(HOST=RAC-SCAN)(PORT=1521))) "+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  "( ADDRESS = (PROTOCOL = TCP)(HOST=DG-SCAN)(PORT=1521)))"+

"(CONNECT_DATA=(SERVICE_NAME=service_name)))");
```

If your application uses Oracle Database 12c or later UCP and does not require an ONS wallet or keystore, the setONSConfiguration method is no longer necessary. Your application can then use ONS auto-configuration.

It is also possible to set ONS configuration using a Java properties file. In this case, the name=value passed to `setONSConfiguration` can only be `propertyfile= <path to ons.properties file>:`

Copyright © 2025, Oracle and/or its affiliates / Public

```
PoolDataSource  pds = PoolDataSourceFactory.getPoolDataSource();


pds.setConnectionPoolName("FCFSamplePool");

pds.setFastConnectionFailoverEnabled(true);

pds.setONSConfiguration("propertiesfile=/usr/ons/ons.properties");

pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");

pds.setURL("jdbc:oracle:thin@((CONNECT_TIMEOUT=90)(RETRY_COUNT=100)"+
" (RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=1000ms)"+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  " ( ADDRESS = (PROTOCOL = TCP)(HOST=RAC-SCAN)(PORT=1521))) "+

" (ADDRESS_LIST = "+

  " (LOAD_BALANCE=on) "+

  "( ADDRESS = (PROTOCOL = TCP)(HOST=DG-SCAN)(PORT=1521)))"+

"(CONNECT_DATA=(SERVICE_NAME=service_name)))");
```

The property file specified must contain an `oracle.ons.nodes` property and optionally, properties for `oracle.ons.walletfile` and `oracle.ons.walletpassword`. An example of an `ons.properties` file is shown here:

```
oracle.ons.nodes=racnode1:4200,racnode2:4200

oracle.ons.walletfile=/oracle11/onswalletfile
```


Copyright © 2025, Oracle and/or its affiliates / Public

Appendix B Troubleshooting FAN

- The following checklist can be used to diagnose FAN delivery and retrieval-related problems: Is one or both Oracle Clusterware Grid Infrastructure or Global Data Services being used? FAN requires Oracle Clusterware or GDS for posting. You can use these with Oracle Restart, RAC, RAC One, DG, and ADG. The important point is that the database is being monitored.
- Is a dynamic database service being used? A dynamic database service is a service created and managed using `srvctl` or `gdsctl`.
- Does the client connect using one of the recommended connect strings discussed in this paper? See examples in the sections titled *General Steps for Configuring FCF clients* or *How to Configure FAN (*for your particular client type).
- Are FAN events being generated at the database tier? Install `FANWatcher` on any database node and confirm FAN events can be generated and received.
- Are FAN events being generated at the client or mid-tier? Install `FANWatcher` on the client or mid-tier node(s) and confirm FAN events can be received.
    - Ensure that the events received at the client or mid-tier are for the DATABASE or SERVICE you are connected to. Examine the FAN event payload for this information.
- Have you set the appropriate settings for FAN on the client side?
    - For Universal Connection Pool with JDBC thin driver, ensure the boolean pool property `FastConnectionFailoverEnabled = true` is set when using Universal Connection Pool with the JDBC thin driver.
    - For ODP.Net, ensure that `pooling=true; HA events=true` is set in the connect string.
    - For OCI clients, refer to the section *How to configure FAN for OCI clients* where you will find examples of how to:
        - Set "`<events>true</events>`" in `oraaccess.xml` and enable notifications on the dynamic database service "`srvctl modify service –notification TRUE` …
    - For WebLogic Active Grid Link, FAN is on by default. This is visible in the admin console. For WebLogic also set ons.configuration at the admin console for the ONS endpoints.
    - For JDBC OCI clients using Universal Connection Pool, set fastConnectionFailover=true
- Check for required patches – WebLogic Active GridLink needs 19033547, 20907322
- You should also check whether your application frequently returns connections to the pool it is using. This is required for Fast Connection Failover functionality, including planned draining, runtime load balancing, and affinity routing, and it is also required by Application Continuity.

Copyright © 2025, Oracle and/or its affiliates / Public

ORACLE

Connect with us

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

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

Copyright © 2025, Oracle and/or its affiliates. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.