# 35
# Transparent Application Failover

This chapter contains the following sections:



## 35.1 Overview of Transparent Application Failover

Transparent Application Failover (TAF) is a feature of the Java Database Connectivity (JDBC) Oracle Call Interface (OCI) driver. It enables the application to automatically reconnect to a database, if the database instance to which the connection is made fails. In this case, the active transactions roll back.

When an instance to which a connection is established fails or is shut down, the connection on the client-side becomes stale and would throw exceptions to the caller trying to use it. TAF enables the application to transparently reconnect to a preconfigured secondary instance, creating a fresh connection, but identical to the connection that was established on the first original instance. That is, the connection properties are the same as that of the earlier connection. This is true regardless of how the connection was lost.

> **Note:**
>
> - TAF is always active and does not have to be set.
> - TAF is not supported with LOB and XML types.

## 35.2 Failover Type Events

The following are possible failover events in the `OracleOCIFailover` interface:

- `FO_SESSION`

  Is equivalent to `FAILOVER_MODE=SESSION` in the `tnsnames.ora` file `CONNECT_DATA` flags. This means that only the user session is authenticated again on the server side, while open cursors in the OCI application need to be reprocessed.

- `FO_SELECT`

  Is equivalent to `FAILOVER_MODE=SELECT` in `tnsnames.ora` file `CONNECT_DATA` flags. This means that not only the user session is re-authenticated on the server side, but open cursors in the OCI can continue fetching. This implies that the client-side logic maintains fetch-state of each open cursor.

- `FO_NONE`

  Is equivalent to `FAILOVER_MODE=NONE` in the `tnsnames.ora` file `CONNECT_DATA` flags. This is the default, in which no failover functionality is used. This can also be explicitly specified to prevent failover from happening. Additionally, `FO_TYPE_UNKNOWN` implies that a bad failover type was returned from the OCI driver.

- `FO_BEGIN`

  Indicates that failover has detected a lost connection and failover is starting.

- `FO_END`

  Indicates successful completion of failover.

- `FO_ABORT`

  Indicates that failover was unsuccessful and there is no option of retrying.

- `FO_REAUTH`

  Indicates that a user handle has been re-authenticated.

- `FO_ERROR`

  Indicates that failover was temporarily unsuccessful, but it gives the application the opportunity to handle the error and retry failover. The usual method of error handling is to issue the `sleep` method and retry by returning the value `FO_RETRY`.

- `FO_RETRY`

  Indicates that the application should retry failover.

- `FO_EVENT_UNKNOWN`

  Indicates a bad failover event.

## 35.3 TAF Callbacks

TAF callbacks are used in the event of the failure of one database connection, and failover to another database connection. TAF callbacks are callbacks that are registered in case of failover. The callback is called during the failover to notify the JDBC application of events generated. The application also has some control of failover.

> **Note:**
>
> The callback setting is optional.

## 35.4 Java TAF Callback Interface

The `OracleOCIFailover` interface includes the `callbackFn` method, supporting the following types and events:

```
public interface OracleOCIFailover{

// Possible Failover Types
public static final int FO_SESSION = 1;
public static final int FO_SELECT  = 2;
public static final int FO_NONE  = 3;
public static final int;
```

```
// Possible Failover events registered with callback
public static final int FO_BEGIN   = 1;
public static final int FO_END     = 2;
public static final int FO_ABORT   = 3;
public static final int FO_REAUTH  = 4;
public static final int FO_ERROR  = 5;
public static final int FO_RETRY  = 6;
public static final int FO_EVENT_UNKNOWN = 7;

public int callbackFn (Connection conn,
                       Object ctxt, // ANy thing the user wants to save
                       int type, // One of the possible Failover Types
                       int event ); // One of the possible Failover Events
```

#### Handling the FO_ERROR Event

In case of an error while failing over to a new connection, the JDBC application is able to retry failover. Typically, the application sleeps for a while and then it retries, either indefinitely or for a limited amount of time, by having the callback return `FO_RETRY`.

#### Handling the FO_ABORT Event

Callback registered should return the `FO_ABORT` event if the `FO_ERROR` event is passed to it.

## 35.5 Comparison of TAF and Fast Connection Failover

Transparent Application Failover (TAF) differs from Fast Connection Failover in the following ways:

- Application-level connection retries

  TAF supports connection retries only at the OCI/Net layer. Fast Connection Failover supports application-level connection retries. This gives the application control of responding to connection failovers. The application can choose whether to retry the connection or to rethrow the exception.

- Integration with the Universal Connection Pool

  TAF works at the network level on a per-connection basis, which means that the connection cache cannot be notified of failures. Fast Connection Failover is well-integrated with the Universal Connection Pool, which enables the Connection Cache Manager to manage the cache for high availability. For example, failed connections are automatically invalidated in the cache.

- Event-based

  Fast Connection Failover is based on the Oracle RAC event mechanism. This means that Fast Connection Failover is efficient and detects failures quickly for both active and inactive connections.

- Load-balancing support

  Fast Connection Failover supports `UP` event load balancing of connections and run-time work request distribution across active Oracle RAC instances.

> **See Also:**
>
> *Oracle Universal Connection Pool for JDBC Developer's Guide*

> **Note:**
>
> Oracle recommends *not* to use TAF and Fast Connection Failover in the same application.