

Dbvisit Standby: Dbvisit Standby: Connect failover to standby database

After a switch-over or fail-over to an Oracle Standby database using Dbvisit Standby, the Oracle Database Administrator (DBA) faces the challenge of switching clients to the new activated standby database which is now running as the new primary database. Previously these clients were connected to the former primary database, and so they must be reconnected to the new primary database.

Dbvisit Standby as well as Oracle Data Guard does not provide a feature to support this.

Generally there are two options for the DBA to achieve this:

1. Change of the database host using either the
 - a) connect string or
 - b) DNS.
2. Use of the NET8 listener fail-over.

1. Change of database host

To begin lets look at option 1. Our examples use the following host and service names:

```
host name primary server: dbprod
host name standby server: dbstandby
service name:             orcl
net service name:         orcl_ha
```

In the case of example 1-a, we are going to change the host name within the TNS connect string after switch-over/fail-over or change the IP from primary server's to standby server's.

The tnsnames.ora entry:

```
orcl_ha=
  (description=
    (address = (host = dbprod) (protocol = tcp) (port = 1521))
    (connect_data = (service_name = orcl))
  )
```

will be changed after the switch-over/fail-over into:

```
orcl_ha=
  (description=
    (address = (host = dbstandby) (protocol = tcp) (port = 1521))
    (connect_data = (service_name = orcl))
  )
```

Or alternatively by using a JDBC-connect:

```
jdbc:oracle:thin:@dbprod:1521/orcl
```

will be changed after the switch-over/fail-over into

```
jdbc:oracle:thin:@dbstandby:1521/orcl
```

This solution seems to be the easiest way if the changes to be made are limited, or on a single point. This type of configuration is mostly found with clients that have a:

- a. centralised tnsnames.ora, or
- b. net service resolution via Oracle Internet Directory or Active Directory, or
- c. any connect string at the Application Server.

This method is only efficient in a centralised names configuration.

The DBA could switch to option 1-b if there is a DNS alias used for the name services. In addition to the host

names defined above we define an extra DNS CNAME as **dbha**, which initially points at the host **dbprod**. The tnsnames.ora as well as the JDBC connect string stay as they are after switch-over/fail-over:

```
orcl_ha=
  (description=
    (address = (host = dbha) (protocol = tcp) (port = 1521))
    (connect_data = (service_name = orcl))
  )
```

or:

```
jdbc:oracle:thin:@dbha:1521/orcl
```

After failover the DNS CNAME **dbha** is redirected to the host name **dbstandby**.

The advantage of this method is that it works well for centralised as well as for the de-centralised names configuration services. The disadvantage with this method is that the connect fail-over can have serious time delays, because the new DNS configuration first has to be replicated to the DNS servers. The name service caches of clients also may need to be flushed. So even though the database might be online and available, some of the clients may still not be able to connect because of the DNS update being delayed.

2. Listener fail-over

The most recommended and most flexible solution is using a dynamic connect listener fail-over. To achieve this the DBA defines **dbprod** as well as **dbstandby** into a common connect string and to establish a fail-over between them both.

The related tnsnames.ora entry would be as follows:

```
orcl_ha=
  (description=
    (address = (host = dbprod) (protocol = tcp) (port = 1521))
    (address = (host = dbstandby) (protocol = tcp) (port = 1521))
    (failover = yes)
    (connect_data = (service_name = orcl))
  )
```

Also for JDBC the DBA would have to use the complete connect string:

```
jdbc:oracle:thin:@(description=(address=(host=dbprod) (protocol=tcp) (port=1521)) (address=(host=dbstandby) (protocol=tcp) (port=1521)) (failover=yes) (connect_data=(service_name=orcl)))
```

In this configuration it's essentially to ensure that only the listener at the present primary server is running. Otherwise it's possible that you run into the problem that the client connections are directed to the standby database, which produces the following error message output:

```
ORA-01033: ORACLE initialisation or shut down in progress
```

If the listener is controlled by either **Oracle Clusterware**, **Oracle Grid Infrastructure** or **Oracle Restart** it would be possible to fully automate the start and shut downs of the listeners. This is done by integrating a control resource at the primary and standby server.

For example, create an action script, which is saved as `/usr/bin/dbv_lsnr_ctrl.sh` and that can be run by the Oracle user:

```
#!/bin/sh
# Listener status control script for use with Dbvisit standby
# - prevents listener from starting if the corresponding instance has standby role
#
unset DEFAULT_DBV_HOME PRIMARY_ROLE_EXPR INSTANCE_NAME
STANDBY_ROLE_EXPR="^Standby Database"
DEFAULT_DBV_HOME=/u01/app/dbvisit
INSTANCE_NAME=${_USR_ORA_SRV}
# set default Dbvisit Standby home path, if not already set in environment
```

```

if [ -z "$DBV_HOME" ]; then
    DBV_HOME=${DEFAULT_DBV_HOME}
fi
case $1 in
    # handle start request and check equally
    # (start/report ok, except when on standby side and instance is running)
    start|check)
        ! ${DBV_HOME}/dbv_oraStartStop status ${INSTANCE_NAME}|grep
        "${STANDBY_ROLE_EXPR}" >/dev/null 2>&1
        exit $?
        ;;
    # stop request always can be fulfilled
    stop)
        exit 0
        ;;
    esac
# paranoia exit
exit 1

```

Within the script (or alternatively at the environment variable DEFAULT_DBV_HOME) the DBA should define the Dbvisit Standby installation path. In addition the following control resource should also be created:

```

crs_profile \
    -create dbv_lsnr_ctrl \
    -t application \
    -a /usr/bin/dbv_lsnr_ctrl.sh \
    -h dbprod1 \
    -p restricted \
    -o as=restore,ci=60,ft=0

```

In directory \$CRS_HOME/crs/public the following .cap file will be created: dbv_lsnr_ctrl.cap.

The database name needs to be entered at line USR_ORA_SRV= exactly the same way that Dbvisit Standby commands are executed:

```
USR_ORA_SRV=orcl (e.g. at "dbvisit orcl")
```

The action script (/usr/bin/dbv_lsnr_ctrl.sh) is called with a parameter which is used to call the Dbvisit Standby command to obtain the database status:

```
dbv_oraStartStop status
```

To add the control resource to **Oracle Clusterware**:

```
crs_register dbv_lsnr_ctrl
```

Finally the dependency of listener and control resource needs to be defined:

```
crs_profile -update ora.LISTENER.lsnr -r dbv_lsnr_ctrl
```

After restarting the listener the dependency on the control resource is created. This determines the status of the database as reported by dbv_oraStartStop, whether it is started or not. If dbv_oraStartStop detects a database in standby role, the control resource goes down and prevents the listener from starting. Otherwise it starts and the listener is allowed to run.

Author: Thilo Solbrig (Oracle Certified Master) www.aspicon.de

¹ On the standby server this should be dbstandby