Requirements for Oracle

Contents

Oracle

Capture	Hub	Integrate
✓	✓	✓

This section describes the requirements, access privileges, and other features of HVR when using Oracle for replication.

For the Capabilities supported by HVR on Oracle, see Capabilities for Oracle.

For information about the supported data types and mapping of data types in source DBMS to the corresponding data types in target DBMS or file format, see Data Type Mapping.

For instructions to quickly set up replication using Oracle, see Quick Start for HVR - Oracle.

Supported Editions

HVR supports the following Oracle editions:

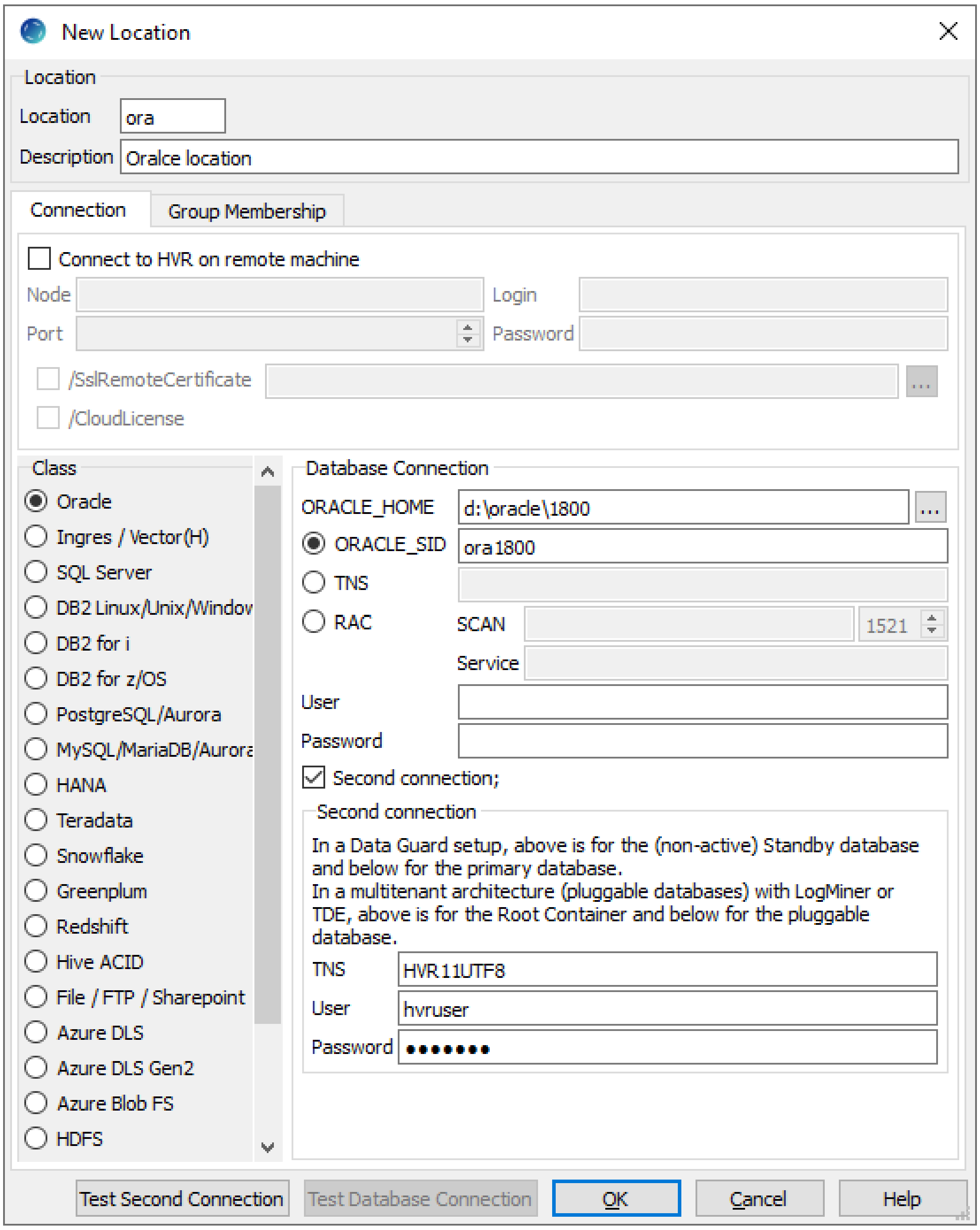
- Enterprise edition
- Standard edition (since HVR 5.6.0/0)

HVR log-based capture is not supported on Oracle 18c Express Edition because the supplemental logging settings cannot be modified in this edition.

For information about compatibility and supported versions of Oracle with HVR platforms, see Platform Compatibility Matrix.

Location Connection

This section lists and describes the connection details required for creating the Oracle location in HVR. HVR uses Oracle's native OCI interface to connect to Oracle.



In a Data Guard setup, above is for the (non-active) Standby database and below for the primary database.
In a multitenant architecture (pluggable databases) with LogMiner or TDE, above is for the Root Container and below for the pluggable database.

Field	Description
Database Connection	
Oracle_Home	The directory path where Oracle is installed. **Example:** D:\oracle
Oracle_SID	The unique name identifier of the instance/database. **Example:** HVR1900
TNS	The connection string for connecting to an Oracle database. The format for the connection string is **host:port/service_name**. Alternatively, the connection details can be added into the clients **tnsnames.ora** file and specify that net service name in this field. **Example:** myserver:1522/HVR1900

RAC	Parameters for connecting to RAC using SCAN configuration. • **SCAN**: Single Client Access Name (SCAN) DNS entry which can be resolved to IP address. • **Service**: The instance **service_name**. **Example:** HVR1900 Ensure that EZCONNECT is specified by the NAMES.DIRECTORY_PATH parameter in the **sqlnet.ora** file. **Example:** NAMES.DIRECTORY_PATH=(ezconnect, tnsnames)
User	The username to connect HVR to the Oracle database. **Example:** redohvruser
Password	The password of the **User** to connect HVR to the Oracle database.
Second connection; either for Data Guard or multitenant architecture (pluggable databases) with LogMiner or TDE	Show/hide **Second Connection**. When this field is selected: • In a Data Guard setup, the above connection parameters/fields are used for connecting to the Standby database and the below connection parameters/fields are used for connecting to the primary database. • In a multitenant architecture (pluggable databases) with LogMiner or TDE, the above connection parameters/fields are used for connecting to the Root container and the below connection parameters/fields are used for connecting to the pluggable database.
TNS	The connection string for connecting to the primary Oracle database. The format for the connection string is **host:port/service_name**. **Example:** myserver:1522/HVR1220
User	The username to connect HVR to the primary Oracle database. **Example:** hvruser
Password	The password of the **User** to connect HVR to the primary Oracle database.

Hub

HVR allows you to create a hub database (schema) in Oracle. The hub database is a repository that HVR uses to control its replication activities. This repository stores HVR catalog tables that hold all specifications of replication such as the names of the replicated databases, the list of replicated tables, and the replication direction.

HVR does not support Oracle containers as hub database.

Grants for Hub Schema

The hub database (schema) can be located inside an Oracle instance on the hub machine, or it can be located on another machine and connected using a TNS connection. The following grants are required for hub database user (e.g. *hvruser*):

```
grant create session to hvruser;
grant create table to hvruser;
grant create trigger to hvruser;
grant create procedure to hvruser;
```

Capture

HVR allows you to Capture changes from an Oracle database. This section describes the configuration requirements for **capturing** changes from the Oracle location. For the list of supported Oracle versions, from which HVR can capture changes, see Capture changes from location in Capabilities.

By default, HVR performs log-based capture.

Table Types

HVR supports capture from the following table types in Oracle:

- Ordinary (heap-organized) tables
- Partitioned tables
- Index-organized tables

Grants for Log-Based Capture

HVR does log-based capture if action **Capture** is defined. HVR can either connect to the database as the owner of the replicated tables, or it can connect as a special user (e.g. *hvruser*).

1. The database **User** must be granted the **create session** privilege.

   ```
   grant create session to hvruser;
   ```

2. To improve the performance of **HVR Initialize** for channels with a large number of tables (more than 150), HVR creates a temporary table (**hvr_sys_table**) within a schema.
   For HVR to automatically create this temporary table the **User** must be granted **create table** privilege.

   If you do not wish to provide **create table** privilege to the **User**, the alternative method is to manually create this temporary table using the SQL statement:

   ```
   create global temporary table hvr_sys_table (table_name varchar
   (128), table_owner varchar(128));
   ```

   This temporary table is not used when capturing from a Data Guard standby database.
3. To replicate tables that are owned by other schemas (using action **TableProperties /Schema**) the **User** must be also granted **select any table** privilege, or the user should be given individual table-level **select** privileges.
4. The database **User** can be granted the **select any dictionary** privilege to read the data dictionaries in Oracle's **SYS** schema.

   ```
   grant select any dictionary to hvruser;
   ```

   Alternatively, the database **User** may be granted **select** privilege only for the required data dictionary objects:

   ```
   grant select on sys.v_$archive_dest to hvruser;

   grant select on sys.v_$archived_log to hvruser;

   grant select on sys.v_$database to hvruser;

   grant select on sys.v_$database_incarnation to hvruser;

   /* The following grant (sys.v_$dnfs_files) is required for
   identifying the redo files located on DirectNFS */

   grant select on sys.v_$dnfs_files to hvruser;
   ```

```
/* The following grant (sys.v_$encryption_wallet) is required for
decryption */

grant select on sys.v_$encryption_wallet to hvruser;

grant select on sys.v_$log to hvruser;

grant select on sys.v_$logfile to hvruser;

/* The following grant (sys.v_$logmnr_contents) is required for
Oracle SQL/LogMiner mode. */

grant select on sys.v_$logmnr_contents to hvruser;

grant select on sys.v_$nls_parameters to hvruser;

grant select on sys.v_$parameter to hvruser;

grant select on sys.v_$pdbs to hvruser;

/* The following grant (sys.v_$system_parameter) is required for
reading the value of 'filesystemio_options' parameter which in turn
is used for reading the redo logs */

grant select on sys.v_$system_parameter to hvruser;

grant select on sys.all_cons_columns to hvruser;

grant select on sys.all_constraints to hvruser;

grant select on sys.all_ind_columns to hvruser;

grant select on sys.all_indexes to hvruser;

grant select on sys.all_lobs to hvruser;

grant select on sys.all_log_groups to hvruser;

grant select on sys.all_objects to hvruser;

grant select on sys.all_tab_cols to hvruser;

grant select on sys.all_tables to hvruser;

grant select on sys.all_triggers to hvruser;

grant select on sys.all_encrypted_columns to hvruser;

grant select on sys.dba_tablespaces to hvruser;

grant select on sys.obj$ to hvruser;

/* The following grant (sys.ecol$) is required for Oracle Database
11.2 and above since default values for added columns are stored
differently. This grant is required since HVR version 5.6.5/3 and
5.6.0/9. */

grant select on sys.ecol$ to hvruser;

grant select on sys.user$ to hvruser;

grant select on sys.col$ to hvruser;

grant select on sys.enc$ to hvruser;

grant select on sys.tabpart$ to hvruser;
grant select on sys.tabsubpart$ to hvruser;

/* The following three grants are required for Refresh (option -qrw)
and action AdaptDDL */

grant select on sys.v_$locked_object to hvruser;

grant select on sys.v_$transaction to hvruser;

grant select on sys.dba_objects to hvruser;
```

5. To capture **create table** statements and add supplemental logging to the newly created table(s), the **User** must be granted the following privilege:

```
grant alter any table to hvruser;
```

6. To use action **DbSequence**, the **User** must be granted the following privileges:

```
grant select any sequence to hvruser;

grant select on sys.seq$ to hvruser;
```

An alternative to all of the above grants is to provide the **sysdba** privilege to the Oracle **User** (e.g. *hvruser*).

- On Unix and Linux, add the user name used by HVR to the line in **/etc/group** for the Oracle sysadmin group.
- On Windows, right-click **My Computer** and select **Manage Local Users and Groups Groups ora_dba Add to Group Add**.

Related Topics	Extra Grants for Supplemental Logging, Extra Grants For Accessing Redo Files Over TNS, Extra Grants for LogMiner, Extra Grants for Amazon RDS for Oracle

Supplemental Logging

HVR needs the Oracle supplemental logging feature enabled on replicate tables that it replicates. Otherwise, when an **update** is done, Oracle will only log the columns which are changed. But HVR also needs other data (e.g. the key columns) so that it can generate a full **update** statement on the target database. The Oracle supplemental logging can be set at the database level and on specific tables. In certain cases, this requirement can be dropped. This requires **ColumnProperties /CaptureFromRowId** to be used and is explained below.

The very first time that **HVR Initialize** runs, it will check if the database allows any supplemental logging at all. If it is not, then **HVR Initialize** will attempt statement **alter database add supplemental log data** (see Extra Grants for Supplemental Logging to execute this statement). Note that this statement will hang if other users are changing tables. This is called 'minimal supplemental logging'; it does not actually cause extra logging; that only happens once supplemental logging is also enabled on a specific table. To see the status of supplemental logging, perform query **select log_group_type from all_log_groups where table_name='mytab'** .

The current state of supplemental logging can be checked with query **select supplemental_log_data_min, supplemental_log_data_pk, supplemental_log_data_all from v$database**. This query should return at least [**'YES'**, **'NO'**, **'NO'**].

Supplemental logging can be easily disabled (**alter database drop supplemental log data**).

HVR Initialize will normally only enable supplemental logging for the key columns of each replicated table, using statement **alter table** *tab1* **add supplemental log data (primary key) columns**. But in some cases, **HVR Initialize** will instead perform **alter table** *tab1* **add supplemental log data (all) columns.** This will happen if the key defined in the replication channel differs from the Oracle table's primary key, or a table has any type of compression enabled, or if one of the following actions is defined on the table:

- On the capture location:
  - **ColumnProperties /CaptureExpression**
  - **Restrict** with **/CaptureCondition** , **/HorizColumn**
  - **TableProperties /DuplicateRows**
- On the target location:
  - **FileFormat /BeforeUpdateColumns**
- On any location:
  - **CollisionDetect**
  - **ColumnProperties** with **/IntegrateExpression** , **/Key** or **/TimeKey**
  - **Integrate** with **/DbProc** or **/Resilient**
  - **Integrate /RenameExpression**
  - **Restrict** with **/AddressTo** or **/IntegrateCondition**

Supplemental Log Data Subset Database Replication

HVR does not support the 'Supplemental Log Data Subset Database Replication' option on Oracle 19c and higher versions. This feature must be disabled for your Oracle database when using HVR for replication.

To verify whether the database is enabled for subset database replication ('YES' or 'NO'), run the following command:

```
select supplemental_log_data_sr from v$database;
```

To disable this option, run the following command:

```
alter database drop supplemental log data subset database replication;
```

Capturing from Oracle ROWID

If none of the above requirements force HVR to enable supplemental logging on all columns, the requirement for supplemental logging on key columns can be removed, if the channel is configured with **ColumnProperties /CaptureFromRowId** and **ColumnProperties /SurrogateKey**. When these actions are defined, HVR will consider the Oracle **rowid** column as part of the table and will use it as the key column during replication, and integrate it into the target database.

The following two additional actions should be defined to the channel to instruct HVR to capture **rowid** v alues and to use them as surrogate replication keys. Note that these actions should be added before adding tables to the channel.

Location	Action	Annotation
Source	**ColumnProperties /Name**=**hvr_rowid /CaptureFromRowId**	This action should be defined for capture locations only.
*	**ColumnProperties /Name**=**hvr_rowid /SurrogateKey**	This action should be defined for both capture and integrate locations.

Extra Grants for Supplemental Logging

The **User** must be granted the privileges mentioned in section Grants for Log-Based Capture and additionally the following grants for using supplemental logging:

1. To execute **alter database add supplemental log data** the **User** must have the **sysdba** privilege. Otherwise, HVR will write an error message which requests that a different user (who does have this privilege) execute this statement.
2. If HVR needs to replicate tables that are owned by other schemas, then optionally the HVR user can also be granted **alter any table** privilege, so that **HVR Initialize** can enable supplemental logging on each of the replicated tables. If this privilege is not granted then **HVR Initialize** will not be able to execute the **alter table…add supplemental log data** statements itself; instead, it will write all the statements that it needs to execute into a file and then write an error message which requests that a different user (who does have this privilege) execute these **alter table** statements.

Accessing Redo and Archive

The Oracle instance must have archiving enabled. If archiving is not enabled, HVR will lose changes if it falls behind the redo logs or it is suspended for a time.

HVR supports capturing changes made by Oracle's Direct-Load INSERT feature (e.g. using **insert** statements with 'append hints' (**insert /*+ append */ into**)). For HVR to capture these changes:

- a table/tablespace must not be in the **NOLOGGING** mode, because in this mode, data is inserted without redo logging.
- the archive log mode must be enabled.

Archiving can be enabled by running the following statement as **sysdba** against a mounted but unopened database: **alter database archivelog**. The current state of archiving can be checked with query **select log_mode from v$database**.

The current archive destination can be checked with the query **select destination, status from v$archive_dest**. By default, this will return values **USE_DB_RECOVERY_FILE_DEST, VALID**, which means that HVR will read changes from within the flashback recovery area. Alternatively, an archive destination can be defined with the following statement: **alter system set log_archive_dest_1='location=/disk1/arch'** and then restart the instance.

Often Oracle's **RMAN** will be configured to delete archive files after a certain time. But if they are deleted too quickly then HVR may fail if it falls behind or it is suspended for a time. This can be resolved either by (a) re-configuring **RMAN** so that archive files are guaranteed to be available for a specific longer period (e.g. 2 days), or by configuring **Hvrlogrelease** . Note that if HVR is restarted it will need to go back to the start oldest transaction that was still open, so if the system has long running transactions then archive files will need to be kept for longer.

Managing Archive/Redo Log files

If log-based capture is defined for an Oracle database (action **Capture**) then HVR may need to go back to reading the Oracle archive/redo files. But each site has an existing backup/recovery regime (normal RMAN) that periodically deletes these Oracle archive files. There are two ways to ensure that these archive files are available for HVR:

- Configure RMAN so that the archive files are always available for sufficient time for the HVR capture job(s). The 'sufficient time' depends on the replication environment; how long could replication be interrupted for, and after what period of time would the system be reset using an HVR Refresh.
- Install command **Hvrlogrelease** on the source machine to make cloned copies of the archive files so that HVR's capture is not affected when these files are purged by the site's backup /recovery regime. When the capture job no longer needs these cloned files, then **Hvrlogrelease** will delete them again.

Capturing from Oracle Data Guard Physical Standby

HVR can perform log-based capture from a Data Guard (DG) physical standby database, as well as from Active Data Guard (ADG) physical database using **DIRECT** capture.

Grants for Capturing from Data Guard Physical Standby Databases

- The HVR user must have a **sysdba** privilege when capturing from a non-Active Data Guard physical standby database.
- To capture from an Active Data Guard physical standby database, the privileges described in Grants for Log-Based Capture apply.

HVR does not support SQL-based capture from a Data Guard physical standby database.

Active Data Guard

To capture from an Active Data Guard physical standby database, the following steps are required:

- Configure the standby database as the HVR location (see below) and define action **Capture** for the location.
- Configure archiving on the standby database.
- Set the necessary log-based capture grants
- Configure supplemental logging on the primary database.

HVR can also capture from an Oracle database that was previously a Data Guard physical database target.

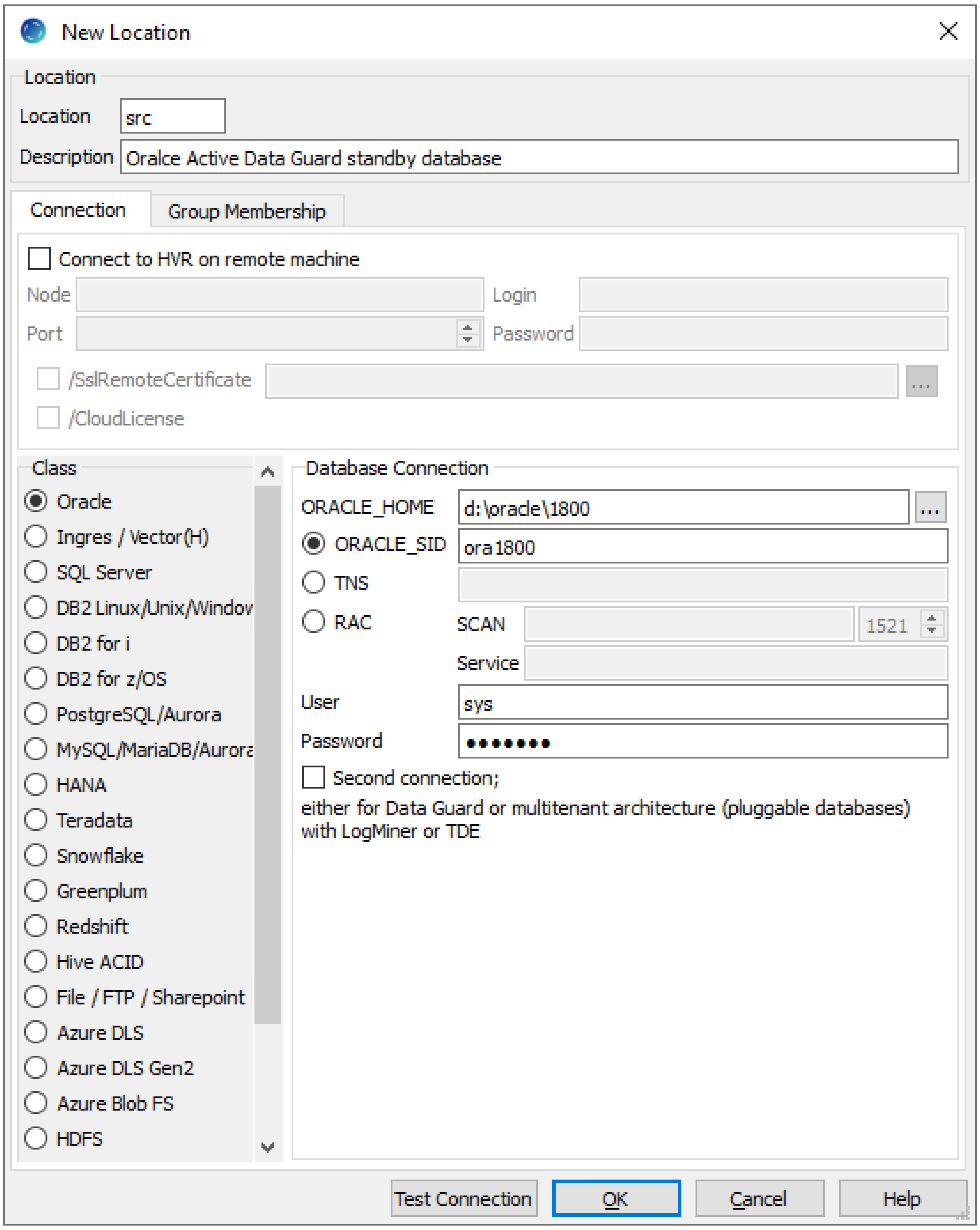
If HVR was capturing changes from one primary Oracle database and a lossless role transition occurs (so that a different Data Guard physical standby database becomes the new primary one), HVR can continue capturing from the new primary, including capturing any changes which occurred before the transition.

This process is automatic providing that the HVR location is connecting to Oracle in a way which 'follows the primary'. When the capture job connects to the database again, it will continue to capture from its last position (which is saved on the hub machine).

Operator intervention is required if a failover took place requiring an OPEN RESETLOGS operation to open the primary database. To start reading transaction logs after OPEN RESETLOGS, perform **HVR Initialize** with option **Table Enrollment**.

Location Connection for Active Data Guard

Following are the connection details required for creating an ADG physical standby database location in HVR:


New Location

Location

Location: src

Description: Oralce Active Data Guard standby database

Connection | Group Membership

Connect to HVR on remote machine [ ]

Node | Login

Port | Password

/SslRemoteCertificate [ ]

/CloudLicense [ ]

Class: [✓ Oracle] Ingres / Vector(H), SQL Server, DB2 Linux/Unix/Window, DB2 for i, DB2 for z/OS, PostgreSQL/Aurora, MySQL/MariaDB/Aurora, HANA, Teradata, Snowflake, Greenplum, Redshift, Hive ACID, File / FTP / Sharepoint, Azure DLS, Azure DLS Gen2, Azure Blob FS, HDFS

Database Connection

ORACLE_HOME: d:\oracle\1800

[✓ ORACLE_SID]: ora1800

TNS

RAC | SCAN | 1521

Service

User: sys

Password: •••••••

Second connection; [ ]
either for Data Guard or multitenant architecture (pluggable databases) with LogMiner or TDE

Test Connection | OK | Cancel | Help

Field	Description
Database Connection	
ORACLE_HOME	The directory path where the standby database is installed.
ORACLE_SID	The unique name identifier of the standby database.
User	The username to connect HVR to the standby database.
Password	The password of the **User** to connect HVR to the standby database.

Non-Active Data Guard

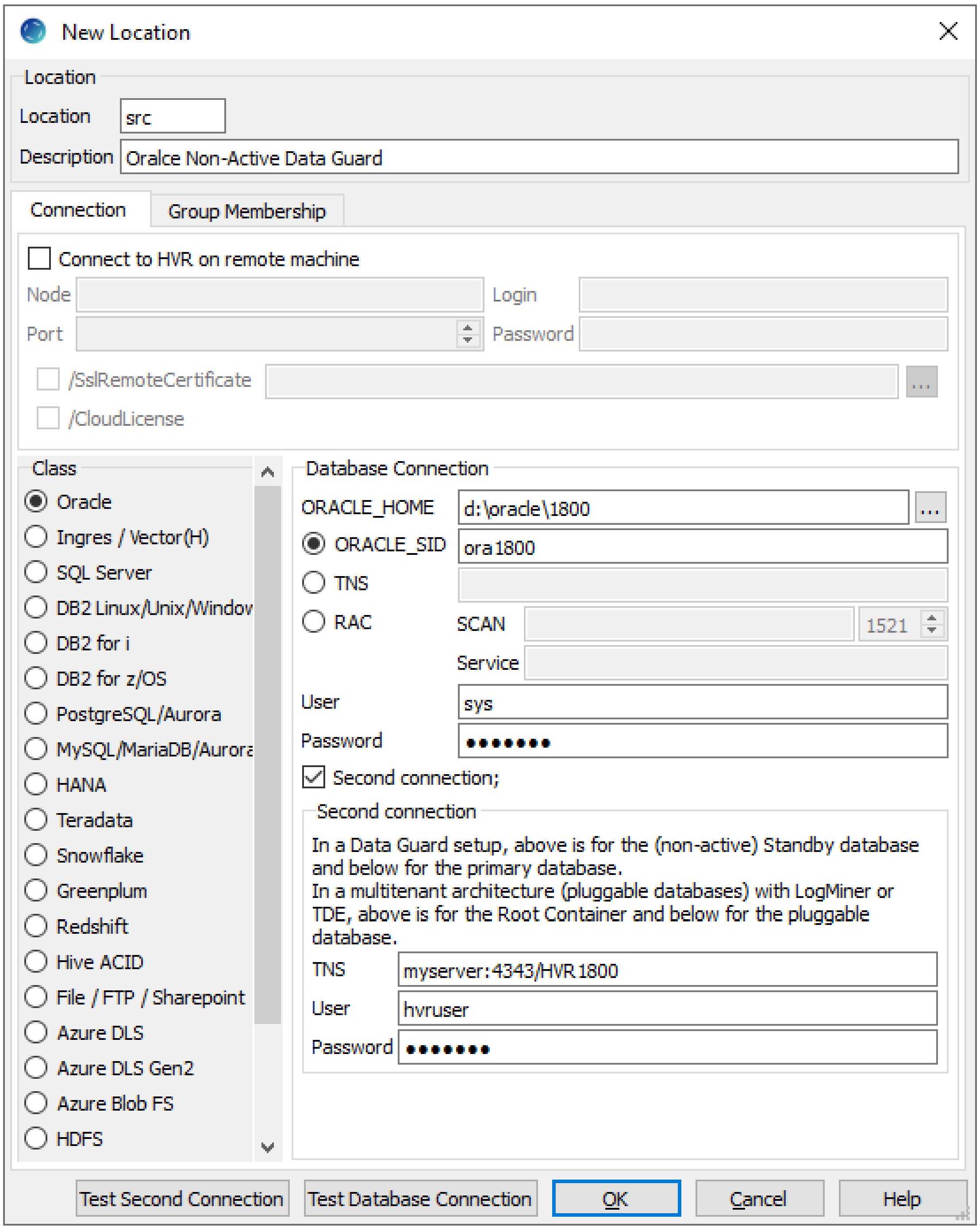
To capture from a non-Active Data Guard physical standby database, the same steps as for Active Data Guard should be done. But the HVR source location must be configured with two connections, one to the physical standby database for access to the redo and one to the primary database to access data (for example, to run HVR Refresh).

In the location creation screen, the lower part ('Primary Connection for selecting data') describes the connection to the primary database. HVR needs a connection to the primary database to do all standard operations (like **HVR Initialize**, **HVR Refresh** and **HVR Compare**). It is assumed that the primary database is reachable from the host of the standby database through a regular TNS connection.

The upper part ('Database Connection') describes the connection to a standby database. HVR needs a connection to the standby database to query the system views about the current state of redo files. With a non-Active Data Guard physical standby database that is in a mounted state as the source, this user must have a **sysdba** privilege.

Location Connection for non-Active Data Guard Physical Standby Database

Following are the connection details required for creating an Oracle Data Guard physical standby database location in HVR:



New Location

Location

Location: src

Description: Oralce Non-Active Data Guard

Connection | Group Membership

☐ Connect to HVR on remote machine

Node: | Login:

Port: | Password:

☐ /SslRemoteCertificate

☐ /CloudLicense

Class

● Oracle
○ Ingres / Vector(H)
○ SQL Server
○ DB2 Linux/Unix/Window
○ DB2 for i
○ DB2 for z/OS
○ PostgreSQL/Aurora
○ MySQL/MariaDB/Aurora
○ HANA
○ Teradata
○ Snowflake
○ Greenplum
○ Redshift
○ Hive ACID
○ File / FTP / Sharepoint
○ Azure DLS
○ Azure DLS Gen2
○ Azure Blob FS
○ HDFS

Database Connection

ORACLE_HOME: d:\oracle\1800

● ORACLE_SID: ora1800

○ TNS

○ RAC   SCAN: 1521

Service:

User: sys

Password: •••••••

☑ Second connection;

Second connection

In a Data Guard setup, above is for the (non-active) Standby database and below for the primary database.
In a multitenant architecture (pluggable databases) with LogMiner or TDE, above is for the Root Container and below for the pluggable database.

TNS: myserver:4343/HVR1800

User: hvruser

Password: •••••••

Test Second Connection | Test Database Connection | OK | Cancel | Help

Field	Description

Database Connection	
ORACLE_HOME	The directory path where a standby database is installed.
ORACLE_SID	The unique name identifier of the standby database.
User	The username to connect HVR to the standby database.
Password	The password of the **User** to connect HVR to the standby database.
Second connection	Show/Hide **Second connection**.
TNS	The connection string for connecting to the primary database. The format for the connection string is **host:port/service_name**.
User	The username to connect HVR to the primary database.
Password	The password of the **User** to connect HVR to the primary database.

Log Read Method - DIRECT (Default)

By default, HVR captures changes using the **DIRECT** log read method (**Capture** **/LogReadMethod** = **DIRECT**). In this method, HVR reads transaction log records directly from the DBMS log file using the file I/O. This method is very fast in capturing changes from the Oracle database. The **DIRECT** log read method requires that the HVR agent is installed on the source database machine.

Oracle parameter **DB_BLOCK_CHECKSUM=OFF** is not supported by log-based capture. Values **TYPICAL** (the default) and **FULL** (unnecessarily high) are supported by HVR.

Extra Grants For Accessing Redo Files Over TNS

For certain Oracle versions (11.2.0.2 and 11.2.0.3), HVR reads the redo and archive files directly through its SQL connection, provided those files are on ASM storage or the connection to the source database is over TNS.

The **User** must be granted the privileges mentioned in section Grants for Log-Based Capture and additionally **select any transaction** privilege for HVR to read the redo and archive files directly through its SQL connection.

Native Access to Redo Files

HVR's capture job needs permission to read Oracle's redo and archive files at the Operating System level. There are three different ways that this can be done;

1. Install HVR so it runs as Oracle's user (e.g. **oracle**).
2. Install HVR under a username (e.g. **hvr**) which is a member of Oracle's default Operating System group (typically either **oinstall** or **dba**).
   - On Unix and Linux the default group of user **oracle** can be seen in the 4th field of its line in **/etc/passwd**. The HVR user be made a member of that group by adding its name to file **/etc/group** (e.g. line **oinstall:x:101:oracle,hvr**).
   - On Windows, right-click **My Computer** and select **Manage** **Local Users and Groups** **Groups** **ora_dba** **Add to Group** **Add**.
     Note that adding HVR's user to group **dba** will also give HVR **sysdba** privilege.
3. Create special Access Control Lists (ACLs) on these files so that HVR's user can read them.

   - On Linux, the following commands can be run as user **oracle** to allow user **hvr** to see redo files in **ORACLE_HOME/oradata/SID** and archive files in **ORACLE_HOME/ora_arch**. Note that an extra "default ACL" is needed for the archive directory, so that future archive files will also inherit the directory's permissions.

```
$ setfacl -R -m u:hvr:rx $ORACLE_HOME/oradata
$ setfacl -R -m u:hvr:rx,d:u:hvr:rx $ORACLE_HOME/ora_arch
```

- On HP UX, the commands are as follows;

```
$ setacl -m u:hvr:rx $ORACLE_HOME/oradata
$ setacl -m u:hvr:rx $ORACLE_HOME/oradata/SID
$ setacl -m u:hvr:rx $ORACLE_HOME/oradata/SID/*
$ setacl -m u:hvr:rx,d:u:hvr:rx $ORACLE_HOME/ora_arch
$ setacl -m u:hvr:rx,d:u:hvr:rx $ORACLE_HOME/ora_arch/*
$ setacl -m u:hvr:rx $ORACLE_HOME/ora_arch/*/*
```

- On Solaris, an extra command is needed to initialize the "default ACL";

```
$ setfacl -m u:hvr:rx $ORACLE_HOME/oradata
$ setfacl -m u:hvr:rx $ORACLE_HOME/oradata/SID
$ setfacl -m u:hvr:rx $ORACLE_HOME/oradata/SID/*
$ setfacl -m d:u::rwx,d:g::rx,d:o:,d:m:rwx $ORACLE_HOME/ora_arch
$ setfacl -m u:hvr:rx,d:u:hvr:rx $ORACLE_HOME/ora_arch
$ setfacl -m u:hvr:rx,d:u:hvr:rx $ORACLE_HOME/ora_arch/*
$ setfacl -m u:hvr:rx $ORACLE_HOME/ora_arch/*/*
```

Sometimes a Unix file system must be mounted in **/etc/fstab** with option **acl** otherwise ACLs are not allowed. On Linux, the user **root** can use command **mount –o remount,acl** to change this dynamically.

Accessing Oracle ASM

HVR supports log-based capture from Oracle databases whose redo and archive files are located on the ASM storage. HVR uses the **dbms_diskgroup** package to access the ASM files for **DIRECT** capture.

To configure this, define environment variable **HVR_ASM_CONNECT** to a username/password pair such as **sys/sys**. The user needs sufficient privileges on the ASM instance; **sysdba** for Oracle version 10 and **sysasm** for Oracle 11+. If the ASM is only reachable through a TNS connection, you can use *username/password@TNS* as the value of **HVR_ASM_CONNECT**. If HVR is not able to get the correct value for the **ORACLE_HOME** of the ASM instance (e.g. by looking into **/etc/oratab**), then that path should be defined with environment variable **HVR_ASM_HOME**. These variables should be configured using environment actions on the Oracle location.

The password can be encrypted using the **hvrcrypt** command. For example:

Unix & Linux

```
$ export HVR_ASM_CONNECT="myuser/`hvrcrypt myuser mypass`"
```

Grants for Capturing from Oracle ASM

The **User** must be granted the following **select** privileges for capturing from Oracle ASM:

The **User** must be granted the privileges mentioned in section Grants for Log-Based Capture and additionally the following grants for capturing from Oracle ASM:

```
grant select on sys.v_$asm_diskgroup_stat to hvruser;

grant select on sys.v_$asm_client to hvruser;
```

Archive Log Only

HVR allows you to capture data from archived redo files in the directory defined using action **Capture/ArchiveLogPath**. It does not read anything from online redo files or the 'primary' archive destination.

This allows the HVR process to reside on a different machine than the Oracle DBMS and read changes from files that are sent to it by some remote file copy mechanism (e.g. **FTP**). The capture job still needs an SQL connection to the database for accessing dictionary tables, but this can be a regular connection.

Replication in this mode can have longer delays in comparison with the 'online' one. To control the delays, it is possible to force Oracle to issue an archive once per predefined period of time.

Since HVR 5.6.0/0, HVR supports cross-platform capture of archived redo files. For example, on a Linux machine where **Capture /ArchivelogOnly** is defined, HVR can capture archived redo files coming from an AIX machine.

On RAC systems, delays are defined by the slowest or the less busy node. This is because archives from all threads have to be merged by SCNs in order to generate replicated data flow.

Capturing Compressed Archive Log Files

Since v5.6.5/4

HVR supports capturing data from compressed archive log files that are moved from a 'primary' archive log directory to a custom directory. HVR automatically detects the compressed files, decompresses them, and reads data from them. This feature is activated when action **Capture /ArchiveLogPath** is set to the custom directory.

HVR supports only **gzip** compressed files.

Archive Log Format

If the names of the compressed archive log files differ from the original names of the archive log files, then action **Capture /ArchiveLogFormat** should be defined to set the relevant naming format. The format variables, such as **%d**, **%r**, **%s**, **%t**, **%z**, supported for Oracle are defined in **ArchiveLogFormat** section on the **Capture** page.

> **Example 1:**
>
> Suppose an archive log file is named '**o1_mf_1_41744_234324343.arc**' according to a certain Oracle archive log format pattern '**o1_mf_<thread>_<sequence>_<some_number>.arc**'. The archive file is copied to some custom directory and compressed to '**o1_mf_1_41744_234324343.arc.gz**' with the **.gz** extension added to its name. In such a case, action **Capture /ArchiveLogFormat** should be defined with the following pattern '**o1_mf_%t_%s_%z.arc.gz**'.
>
> **Example 2:**
>
> Suppose the compressed archive log file is named **CQ1arch1_142657_1019376160.dbf.Z** with the **.Z** extension added to its name. In such a case, action **Capture /ArchiveLogFormat** should be defined with the following pattern '**CQ1arch%t_%s_%r.dbf.Z**'.

If action **Capture /ArchiveLogFormat** is not defined, then by default, HVR will query the database for Oracle's initialization parameter - **LOG_ARCHIVE_FORMAT**. The following are used by HVR if action **Capture /ArchiveLogFormat** is not defined,

- For Oracle ASM system, the default name pattern used is '**thread_%t_seq_%s.%d.%d**'.
- Non-ASM system,
  - if Fast-Recovery-Area (FRA) is used, then the default name pattern used is '**o1_mf_%t_%s_%z_.arc**'
  - if FRA is not used, then HVR uses the following SQL query:

```
SELECT value

FROM v$parameter

WHERE name = 'log_archive_format'
```

HVR picks up the first valid archive destination and then finds the format as described above.

To determine whether the destination uses FRA, HVR uses the following query:

```
SELECT destination

FROM v$archive_dest

WHERE dest_name='LOG_ARCHIVE_DEST_[n]';
```

For example, for destination 1, the query is as follows:

```
SELECT destination

FROM v$archive_dest

WHERE dest_name='LOG_ARCHIVE_DEST_1';
```

If the query returns **USE_DB_RECOVERY_FILE_DEST**, it indicates the destination uses FRA.

Capturing Encrypted (TDE) Tables

HVR supports capturing tables that are encrypted using Oracle Transparent Data Encryption (TDE). Capturing tables located in encrypted tablespace and tables with encrypted columns are supported for Oracle version 11 and higher.

HVR supports software and hardware (HSM) wallets. If the wallet is not configured as auto-login (Oracle internal file **cwallet.sso**), using command **Hvrlivewallet** set the password for the wallet on HVR Live Wallet port.

Software wallets can be located in ASM or in a local file system. If the wallet is located in a local file system then HVR either needs permission to read the wallet file or an HVR trusted executable should be created in the directory **HVR_HOME/sbin** with **chmod +4750** . If the wallet located in a local file system is configured as auto-login, then HVR or the trusted executable must be run as the user who created the wallet (usually the **oracle** user).

In Oracle 12, for replicating encrypted columns, *hvruser* should have explicit **select** privileges on **sys.user$** and **sys.enc$** tables.

```
grant select on sys.user$ to hvruser;
grant select on sys.enc$ to hvruser;
```

Further channel configuration changes are not required; HVR automatically detects encryption and opens the wallet when it is encountered.

HVR does not support capturing encrypted (TDE) tables on the HP-UX platform.

Log Read Method - SQL (LogMiner)

HVR captures changes using the **SQL** log read method (**Capture /LogReadMethod** = **SQL**). In this method, HVR uses dbms_logmnr package to read transaction log records using a special SQL function. This method reads change data over an SQL connection and does not require the HVR agent to be installed on the source database machine. However, the **SQL** log read method is slower than the **DIRECT** log read method and exposes additional load on the source database.

The **SQL** log read method enables capture from LogMiner.

Limitations of SQL Log Read Method

- Only Oracle version 11.2.0.3 and above are supported for capturing changes from LogMiner.
- Updates that only change LOB columns are not supported.
- Capture from XML Data Type columns is not supported.
- Index Organized Tables (IOT) with an overflow segment is not supported.
- Capturing DDL (using **AdaptDDL**) changes such as **add table as...** , **drop table...** and **alter table...**, including partition operations are not supported.
- Capture of truncate statements is not supported.

Extra Grants for LogMiner

The **User** must be granted the privileges mentioned in section Grants for Log-Based Capture and additionally the following grants for using LogMiner:

1. **execute on dbms_logmnr**
2. **select any transaction**
3. **execute_catalog_role**
4. For Oracle 12.1 and later, the **User** must be granted the **logmining** system privilege.

Related Topics	Extra Grants for Amazon RDS for Oracle

Amazon RDS for Oracle

Since v5.3.1/9

HVR supports log-based capture and integrate into Amazon RDS for Oracle database. This section provides the information required for replicating changes in Amazon RDS for Oracle.

The following logging modes must be enabled for the Amazon RDS DB instances. You can use the Amazon RDS procedure mentioned below to enable/disable the logging modes. For more information, see Common DBA log tasks for Oracle DB instances in AWS Documentation.

- **Force Logging** - Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments (NOLOGGING clauses are ignored).

  ```
  exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
  ```

- **Supplemental Logging** - To ensure that LogMiner and products that use LogMiner have sufficient information to support chained rows and storage arrangements such as cluster tables.

  ```
  exec rdsadmin.rdsadmin_util.alter_supplemental_logging(p_action =>'AD
  D');
  ```

- **Switch Online Log Files** - To prevent the following error in HVR: Log scanning error F_JZ1533. The scanner could not locate the RBA file sequence *number* in the thread.

  ```
  exec rdsadmin.rdsadmin_util.switch_logfile;
  ```

- **Retaining Archive Redo Logs** - To retain archived redo logs on your Amazon RDS DB instance, database backups must be enabled by setting the archivelog retention hours to greater than 0 (zero) hours. Enabling database backup can be done while creating the instance or after by going to **Instances > Modify > Backup** and set the number of days.

  The following example retains 24 hours of the redo log.

  ```
  begin
  rdsadmin.rdsadmin_util.set_configuration (name => 'archivelog
  retention hours', value => '24');
  end;
  /
  ```

  Set the backup retention period for your Amazon RDS DB instance to one day or longer. Setting the backup retention period ensures that the database is running in ARCHIVELOG mode.

  For example, enable automated backups by setting the backup retention period to three days:

  ```
  aws rds modify-db-instance \

  --db-instance-identifier mydbinstance \

  --backup-retention-period 3 \

  --apply-immediately
  ```

  In Amazon RDS for Oracle, disabling automated backups may lead to replication issues in HVR.

For better performance, it is recommended to install HVR on Amazon Elastic Cloud 2 (EC2) instance in the same region of the RDS instance. For more information about installing the HVR image on AWS, see Installing HVR on AWS using HVR Image.

Extra Grants for Amazon RDS for Oracle

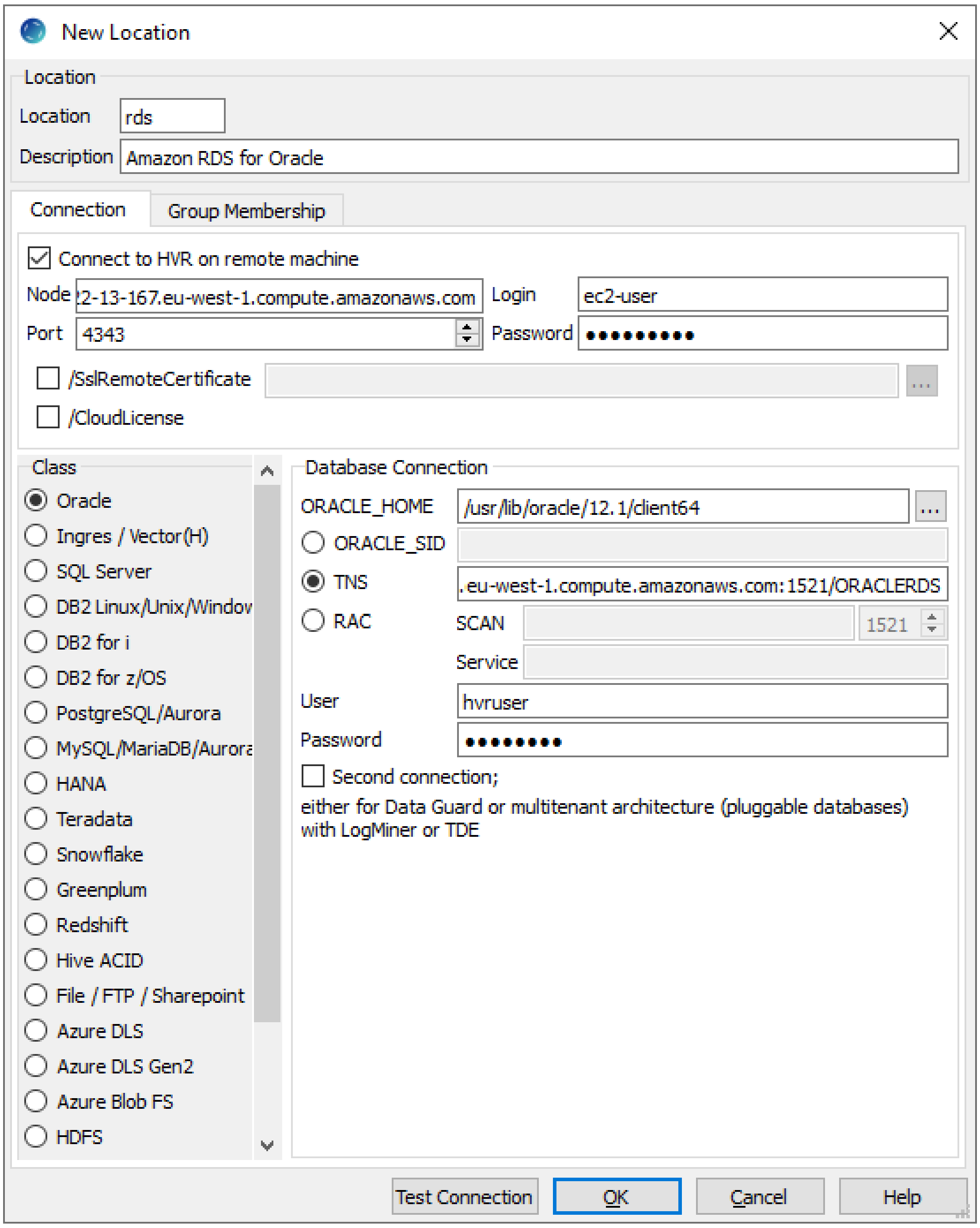
The **User** must be granted the privileges mentioned in sections Grants for Log-Based Capture, Extra Grants for LogMiner and additionally the following grants for using Amazon RDS for Oracle:

1. The **User** must be granted the **select any transaction** privilege.
2. For Amazon RDS for Oracle 12 and later, the **User** must be granted the **logmining** system privilege.
3. The **User** must be granted the LogMiner specific grants by using the Amazon RDS procedure. For more information, refer to Common DBA System Tasks for Oracle DB Instances in AWS Documentation.

   ```
   begin
   rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name     => 'DBMS_LOGMNR',
    p_grantee      => 'HVRUSER',
    p_privilege    => 'EXECUTE',
    p_grant_option =>  true);
   end;
   /
   ```

Location Connection - Amazon RDS for Oracle

Following are the connection details required for creating an Amazon RDS for Oracle location in HVR:


New Location

Location

Location: rds

Description: Amazon RDS for Oracle

Connection | Group Membership

[✓] Connect to HVR on remote machine

Node: :2-13-167.eu-west-1.compute.amazonaws.com
Login: ec2-user

Port: 4343
Password: ●●●●●●●●●

[ ] /SslRemoteCertificate

[ ] /CloudLicense

Class
- [● Oracle]
- Ingres / Vector(H)
- SQL Server
- DB2 Linux/Unix/Windov
- DB2 for i
- DB2 for z/OS
- PostgreSQL/Aurora
- MySQL/MariaDB/Aurora
- HANA
- Teradata
- Snowflake
- Greenplum
- Redshift
- Hive ACID
- File / FTP / Sharepoint
- Azure DLS
- Azure DLS Gen2
- Azure Blob FS
- HDFS

Database Connection

ORACLE_HOME: /usr/lib/oracle/12.1/client64

ORACLE_SID [● TNS] RAC

TNS: .eu-west-1.compute.amazonaws.com:1521/ORACLERDS

SCAN: 1521

Service

User: hvruser

Password: ●●●●●●●●

[ ] Second connection;
either for Data Guard or multitenant architecture (pluggable databases) with LogMiner or TDE

Field	Description
Connect to HVR on remote machine	
Node	The Public DNS of the EC2 instance.
Port	The port for the **HVR Remote Listener**.
Login	The operating system username for the EC2 instance of the **HVR Remote Listener**.
Password	The password for the operating system user account. This password is not validated if the **HVR Remote Listener** is started without password validation (option **-N**).
SslRemoteCertificate	The SSL certificate created on EC2 instance.

Database Connection	
ORACLE_HOME	The ORACLE HOME path of the EC2 instance. **Example:** /usr/lib/oracle/12.1/client64
TNS	The connection string for connecting to the RDS database. The format for the connection string is **AWS Endpoint:Port/DB Name**. Alternatively, the connection details can be added into the client's **tnsnames.ora** fi le and specify that net service name in this field.
User	The username to connect HVR to the Amazon RDS for Oracle database. Example: hvruser
Password	The password of the **User** to connect HVR to the Amazon RDS for Oracle database.

Capturing from Amazon RDS for Oracle

HVR uses LogMiner to capture from Amazon RDS for Oracle. DDL changes are not captured since LogMiner is used for capture. To Capture from Amazon RDS for Oracle, the following action definitions are required:

Group	Table	Action
AMAZONRDSORACLE	*	**Capture /LogReadMethod** = **SQL**
AMAZONRDSORACLE	*	**Environment /Name** = **TZ /Value** = **UTC**

Capturing from Oracle RAC

When capturing from Oracle Real Application Clusters (RAC), HVR will typically connect with its own protocol to an **HVR Remote Listener** installed in the RAC nodes. **HVR Remote Listener** should be configured to run inside all cluster nodes using the same Port, User/Login, and Password for all the Nodes. The hub then connects to one of the remote nodes by first interacting with the Oracle RAC 'SCAN' address.

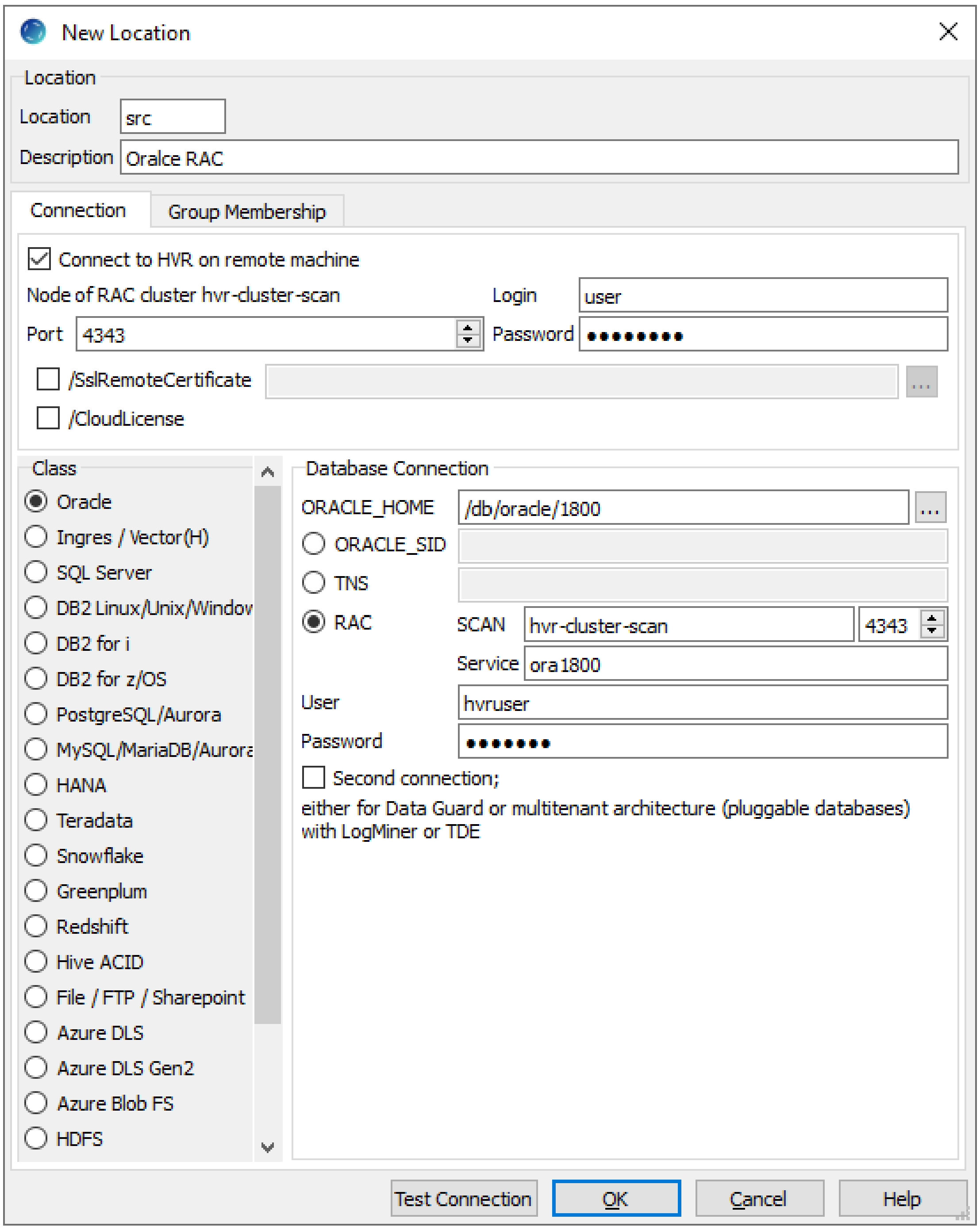
The HVR channel only needs one location for the RAC and there is only one capture job at runtime. This capture job connects to just one node and keeps reading changes from the shared redo archives for all nodes.

Directory **HVR_HOME** and **HVR_CONFIG** should exist on both machines but does not normally need to be shared. If **HVR_TMP** is defined, then it should not be shared.

Prior to HVR 5.5.0/8, capture from Oracle RAC was only supported using the **DIRECT** mode (**Capture / LogReadMethod** = **DIRECT**). However, since HVR 5.5.0/8, capture from Oracle RAC is also supported using the **SQL** mode (**Capture /LogReadMethod** = **SQL**).

Location Connection for Oracle RAC

Following are the connection details required for creating an Oracle RAC location in HVR:



New Location

Location

Location: src

Description: Oralce RAC

Connection | Group Membership

[✓] Connect to HVR on remote machine

Node of RAC cluster hvr-cluster-scan

Port: 4343

Login: user

Password: ●●●●●●●●

[ ] /SslRemoteCertificate

[ ] /CloudLicense

Class

[✓ Oracle] Ingres / Vector(H) SQL Server DB2 Linux/Unix/Windov DB2 for i DB2 for z/OS PostgreSQL/Aurora MySQL/MariaDB/Aurora HANA Teradata Snowflake Greenplum Redshift Hive ACID File / FTP / Sharepoint Azure DLS Azure DLS Gen2 Azure Blob FS HDFS

Database Connection

ORACLE_HOME: /db/oracle/1800

ORACLE_SID TNS [✓ RAC]

SCAN: hvr-cluster-scan 4343

Service: ora1800

User: hvruser

Password: ●●●●●●●

[ ] Second connection;

either for Data Guard or multitenant architecture (pluggable databases) with LogMiner or TDE

Field	Description
Connect to HVR on remote machine	
Port	The TCP/IP port number of the **HVR Remote Listener** on the RAC nodes.
Login	The operating system username for the RAC nodes where the **HVR Remote Listener** is running.
Password	The password for the operating system user account.
Database Connection	
ORACLE_HOME	The directory path where an Oracle RAC database is installed.

SCAN	The Single Client Access Name (SCAN) DNS entry which can be resolved to IP address. **Example: hvr-cluster-scan**
Service	The Oracle service name. **Example: HVR1900**
User	The username to connect HVR to the Oracle RAC database.
Password	The password of the **User** to connect HVR to the Oracle RAC database.

Capturing from Oracle Pluggable Database

HVR supports capturing from Oracle's pluggable database. A pluggable database (PDB) is a portable collection of schemas, schema objects, and non-schema objects that appears to an Oracle Net client as a separate database. One or more PDBs together function as a root multitenant container database (CDB). CDB is where the PDB is managed. For more information about PDB and CDB, refer to Oracle's documentation.

HVR supports both **DIRECT** and **SQL** log read method for capturing from PDB.

- For capturing from a PDB using **DIRECT** method, a single connection to the PDB is sufficient. The connection string for the PDB should point to the database service of the PDB.
- For capturing from a PDB using **SQL** method, two underlying connections must be set up for the capture location: one to a PDB and other one to a CDB, to which the PDB is plugged. For the PDB, the connection string should point to the database service of the PDB. The connection method for PDB is always **TNS**.

Grants for Capturing from Pluggable Database

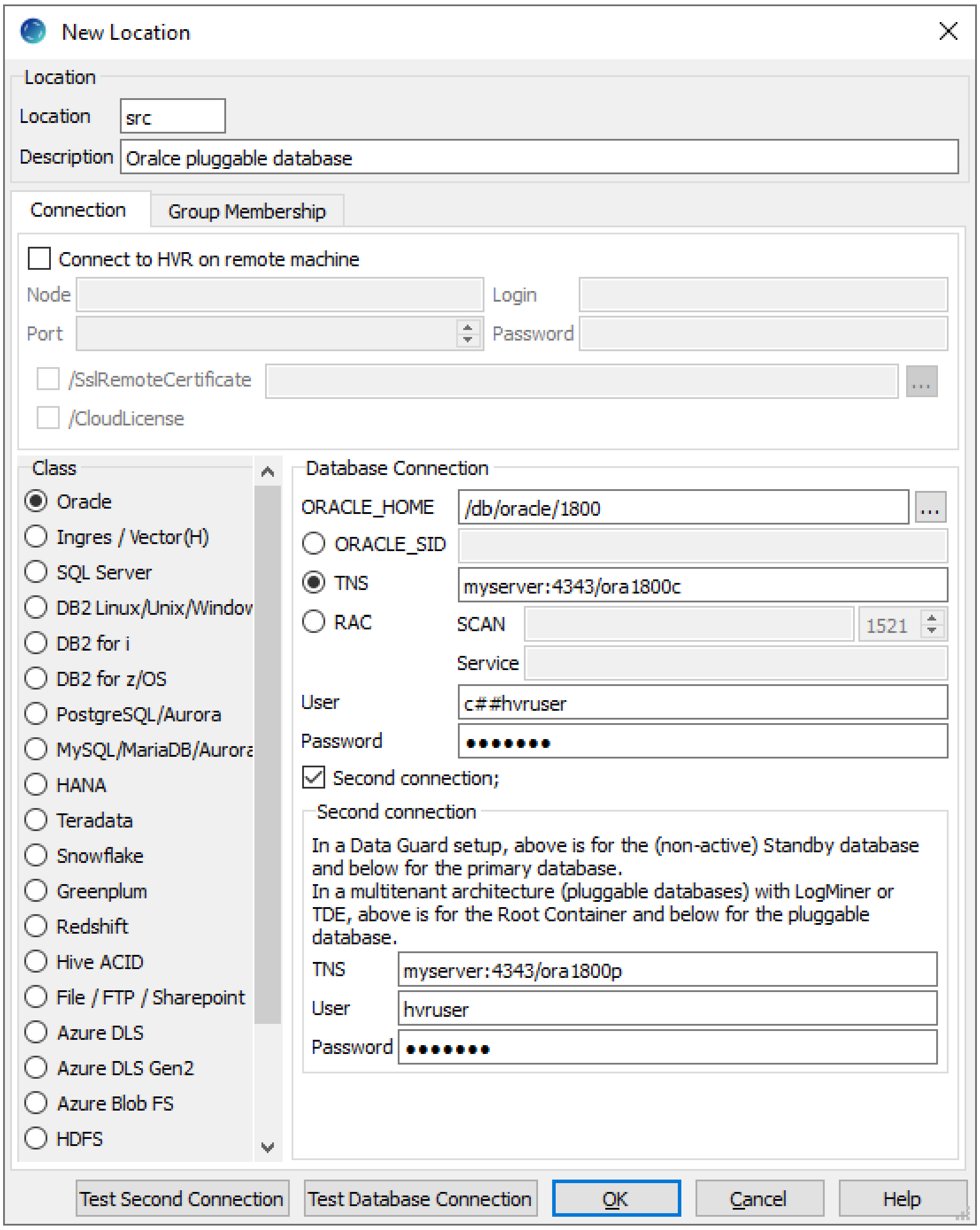
The container database **User** must be a CDB common user with the default prefix **c##** (for example, **c##hvruser**).

The privileges required for the CDB common user (**User**) and the PDB user are the same as the log-based capture grants. For granting privileges to the PDB user, the following command should be first executed to switch to that container (PDB user):

```
ALTER SESSION SET CONTAINER=pdbuser;
```

Location Connection for Pluggable Database

Following are the connection details required for creating an Oracle pluggable database location in HVR:



Location: src

Description: Oralce pluggable database

[✓ Oracle]

ORACLE_HOME: /db/oracle/1800

[✓ TNS]: myserver:4343/ora1800c

SCAN port: 1521

User: c##hvruser

Password: •••••••

[✓] Second connection;

In a Data Guard setup, above is for the (non-active) Standby database and below for the primary database.
In a multitenant architecture (pluggable databases) with LogMiner or TDE, above is for the Root Container and below for the pluggable database.

TNS: myserver:4343/ora1800p

User: hvruser

Password: •••••••

Field	Description
Database Connection	
ORACLE_HOME	The directory path where a CDB is installed.
TNS	The connection string required for connecting to the CDB (where the PDB is located). The format for the connection string is **host:port/service_name**.
User	The username to connect HVR to the CDB. **Example:** c##hvruser
Password	The password of the **User** to connect HVR to the CDB.
Second Connection	Show/Hide **Second Connection**.

TNS	The connection string to a PDB. All allowed TNS connection methods are supported. For example, **host:port/service_name**.
User	The username to connect HVR to the PDB. **Example:** hvruser
Password	The password of the **User** to connect HVR to the PDB.

Trigger-Based Capture

HVR allows you to perform trigger-based capture when action **Capture** is defined with parameter **/TriggerBased**. HVR can either connect to the database as the owner of the replicated tables, or it can connect as a special user (e.g. **hvr**).

Grants for Trigger-Based Capture

1. The database **User** must have the following privileges:
   - **create session**
   - **create table**
   - **create trigger**
   - **create sequence**
2. To replicate tables which are owned by other schemas (using action **TableProperties** **/Schema**) the **User** must be granted the following privileges :
   - **select any table**
   - **execute any procedure**
   - **create any trigger**
3. To read the data dictionaries in Oracle's **SYS** schema the **User** must be granted **select any dictionary** privilege.
   An alternative to this specific grant is to provide the **sysdba** privilege to **User**.
4. Trigger-based capture will use package **dbms_alert** , unless action **Capture** **/TriggerBased** is defined with parameter **/ToggleFrequency** or action **Scheduling** is defined with parameters **/CaptureStartTimes** or **/CaptureOnceOnStart** . This grant can only be given by a user with the **sysdba** privilege.

Upgrading Oracle Database on Source Location

When upgrading your Oracle source database to a next release version, e.g. from 11g to 12c, the compatible mode can still be set to 11g.

The best practice when upgrading an Oracle source database to ensure no data is lost would be as follows:

1. Stop the application making changes to the database.
2. Ensure all the changes made by the application are captured: anticipate the latency to be at zero. For more information on monitoring the replication latency, refer to the **Statistics** page.
3. Stop all capture and integrate jobs under the **HVR Scheduler**.
4. Upgrade the database.
5. Run **HVR Initialize** with the following options selected: **Transaction Files and Capture Time**, **Table Enrollment**, and **Scripts and Jobs**.
6. Restart all the jobs under the **HVR Scheduler**.
7. Start the application.

Integrate and Refresh Target

HVR allows you to **Integrate** or **HVR Refresh** changes into Oracle database in the target location. This section describes the configuration requirements for integrating changes (using **Integrate** and **Refresh**) into Oracle location. For the list of supported Oracle versions into which HVR can integrate changes, see Integrate changes into location in Capabilities.

HVR uses the following interfaces to write data to Oracle during **Integrate** and **Refresh**:

- Oracle native OCI interface, used for continuous **Integrate** and row-wise **Refresh**.
- Oracle OCI direct-path-load interface, used for **Integrate** with **/Burst** and Bulk **Refresh.**

Grants for Integrate and Refresh

1. To **Integrate** changes into a database, or to load data into a database using **HVR Refresh**, the **User** must be granted the following privileges:
   - **create session**
   - **create table**
2. The **User** must be allocated a quota on the default tablespace. For this, the **User** must be granted **alter user ... quota ... on ...** or **grant unlimited tablespace to ...** privilege.
3. To change tables which are owned by other schemas (using action **TableProperties /Schema**) the **User** must be granted the following privileges:
   - **select any table**
   - **insert any table**
   - **update any table**
   - **delete any table**
4. To perform **bulk refresh** (option **-gb**) of tables which are owned by other schemas, the **User** must be granted the following privileges :
   - **alter any table**
   - **lock any table**
   - **drop any table** (needed for **truncate** statements)
5. To disable/re-enable triggers in target schema the **User** must be granted **alter any trigger** and **create any trigger** privilege.
6. If **HVR Refresh** will be used to create target tables then the **User** must be granted the following privileges:
   - **create any table**
   - **create any index**
   - **drop any index**
   - **alter any table**
   - **drop any table**
7. If action **Integrate /DbProc** is defined, then the **User** must be granted **create procedure** privilege.
8. If action **DbSequence /Schema** is defined then the **User** must be granted the following privileges:
   - **create any sequence**
   - **drop any sequence**

Compare and Refresh Source

HVR allows you to perform **HVR Compare** and **HVR Refresh** for Oracle database in the source location.

Grants for Compare or Refresh (Source Database)

1. To perform **HVR Compare** and **HVR Refresh**, the **User** must be granted the **create session** privilege.
2. If **HVR Compare** or **HVR Refresh** needs to read from tables which are owned by other schemas (using action **TableProperties /Schema**) the **User** must be granted **select any table** privilege.
3. If the **Select Moment** feature (option **-M** in **HVR Compare** and **HVR Refresh**) is used then the **User** must be granted the following privileges:
   - **flashback any table**
   - **select any transaction**