Hvrinit

Contents

Name

hvrinit - Load a replication channel.

Synopsis

hvrinit [*options*] *hubdb chn*

Description

Command **hvrinit** encapsulates all steps required to generate and load the various objects needed to enable replication of channel *chn*. These objects include replication jobs and scripts as well as database triggers/rules for trigger-based capture and table enrollment information for log-based capture. For more information about objects, see section Objects below. This command also allows you to perform capture rewind. For more information about capture rewind, see section Capture Rewind below.

The argument *hubdb* specifies the connection to the hub database. For more information about supported hub databases and the syntax for using this argument, see Calling HVR on the Command Line.

Create or Replace Objects or Drop Objects

By default, for the command **hvrinit** or in the **HVR Initialize** dialog, **Create or Replace Objects** is selected resulting in objects to be created or recreated and the system to be initialized.

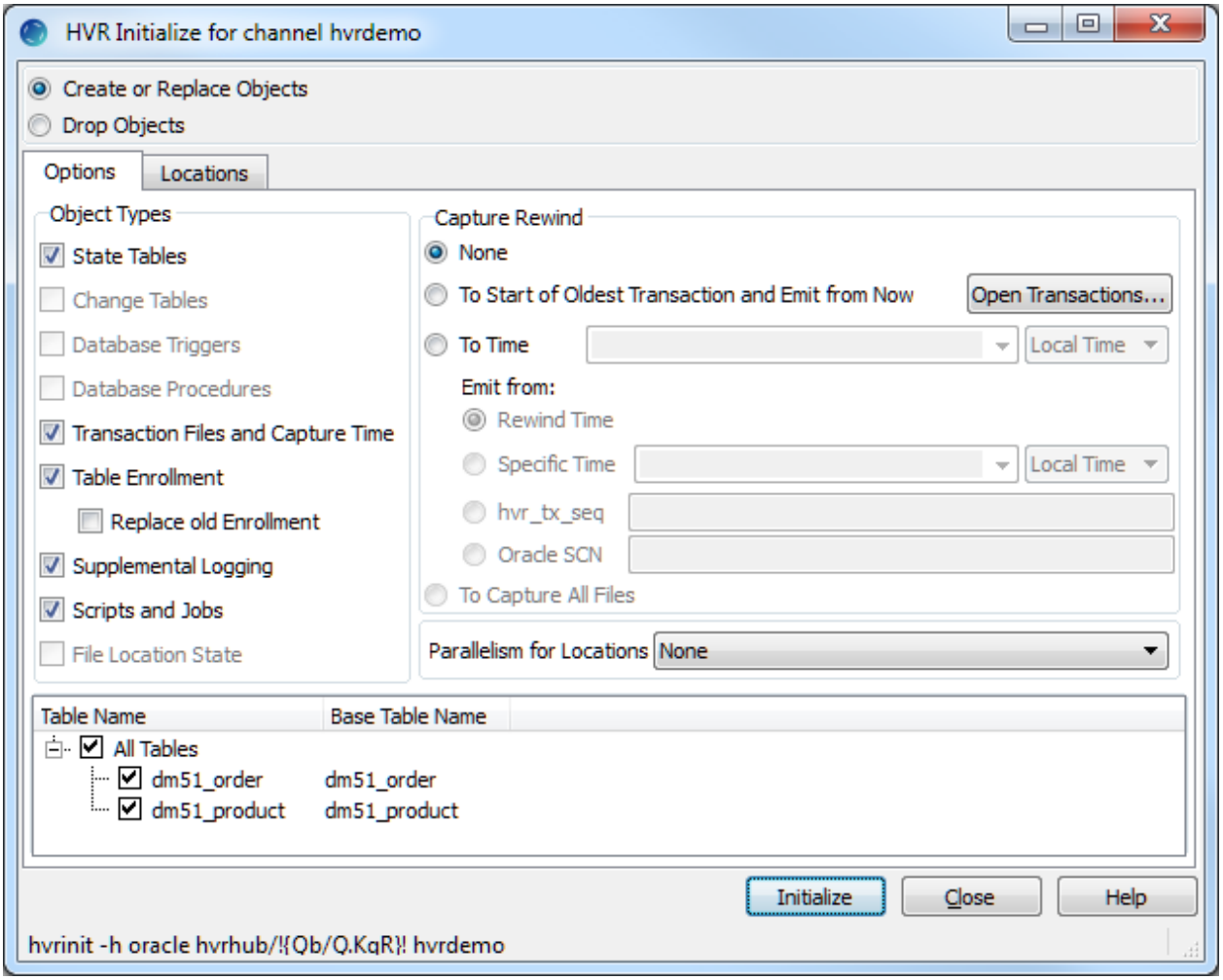
However, objects can be dropped in HVR Initialize dialog by selecting **Drop Objects** (command option **-d**). Use the option **Drop Objects** to remove a channel or a location out of an environment knowing that objects will be dropped only based on the current definition of the channel. For example, if collision history tables were created due to the **CollisionDetect** action then these tables will only be dropped if the **CollisionDetect** action is still part of the current channel definition. It is always possible to manually cleanup an environment if a channel should be removed based on generated database object names and folder structures. For more information about object names, see **Naming of HVR Objects Inside Database Locations**.

Supplemental log groups on an Oracle capture location will not be dropped because HVR makes no assumptions about whether other tools may be taking advantage of the supplemental logging that it may or may not have put in place.

On a SQL Server capture location CDC tables that were created by HVR will be dropped.

Options

This section describes the options available for command **hvrinit**.



Parameter	Description
-d	Drop objects only. If this option is not supplied, then **hvrinit** will drop and recreate the objects associated with the channel such as HVR scripts, internal tables and any transaction files containing data in the replication pipeline. Only a few objects are preserved such as job groups in the scheduler catalogs; these can be removed using **hvrinit -d**.
-E Since v5.3.1/1	Recreates (replace) enroll file for all tables present in the channel. In HVR versions released between 5.3.1/5 and 5.5.0/2, enroll file is recreated only for the tables that are selected during **hvrinit**. Using **hvrinit -E** is same as **hvrinit -osctprEljf** (in HVRGUI it is same as selecting all options under **Object Types**).
-h*class*	Location *class* of the hub database. Valid values for *class* are **db2**, **db2i**, **ingres**, **mysql**, **oracle**, **postgresql**, **sqlserver**, or **teradata**. For more information, see Calling HVR on the Command Line.
-i*x*	Capture rewind. Initialize channel to start capturing changes from a specific time in the past, rather than only changes made from the moment the **hvrinit** command is run. Capture rewind is supported for most databases with log-based capture (not for trigger-based capture i.e. **/TriggerBased** parameter) and for capture from file locations when parameter **/DeleteAfterCapture** is not defined.

Values of *x* may be one of the following:

- *oldest_tx*: Capture changes from the beginning of the oldest current (not closed) transaction (transactions that do not affect channel's tables will not be considered) and emit from **now**. In HVR GUI, this option is displayed as **To Start of Oldest Transaction and Emit from Now**.
- *time*: Start capturing changes from a specific time in the past. This option may be used with option **-I** to define emit start time. If emit time (option **-I**) is not specified, then emit time will be same as capture *time*. In HVR GUI, this option is displayed as **To Time**.
  Valid formats for *time* are *YYYY-MM-DD [HH:MM:SS]* (in local time) or *YYYY-MM-DD***T***HH:MM:SS+TZD* or *YYYY-MM-DD***T***HH:MM:SSZ* or **today** or **now***[±SECS]* or an integer (seconds since **1970-01-01 00:00:00 UTC**).
  For example, **-i "2017-11-01 12:52:46"** or **-i 2017-11-01T12:52:46-06:30** or **-i 2017-11-01T12:52:46Z** or **-i now-3600** (for one hour ago).
- *min*: Capture all available changes in file location.
- **integ_recov_all**: Use existing integrate state tables to decide the rewind and emit start time for capture. This option is meant to be used in a hub failover scenario. The channel definition before and after the failure should be identical (action definition and tables) for this option to work successfully. This option uses data from integrate state table filled by the last integrate job.
  Note that initializing target location(s) with option state tables will destroy the state in the state tables. For this reason you must initialize the capture time separately without the state tables. Upon such failover it is recommended to not re-initialize with option state tables. This option is available only in CLI. **Since** v5.7.5/8
- **integ_recov=***integloc*[**:***oldchn***-cap-***oldcaploc*][**:***oldchn***-integ-***oldtgtloc*]... : This option is similar to **integ_recov_all**. However, it has more advanced customization options to select a specific target location(s). Also, with this option, rewind is possible in case the name of the channel or source or target location(s) have been changed in the new setup. This option is available only in CLI. **Since** v5.7.5/8

  Following are a few examples/scenarios for using this option:
  Let us assume, before recovery, the name of the channel was **channel1**, source location was **src1**, and target location was **tgt1**.

  - If there are multiple target locations, you can also supply more than one **-i integ_recov** with **hvrinit** command. For example, if there are three target locations - **tgt1**, **tgt2**, and **tgt3**, then you can use **hvrinit -i integ_recov=tgt1 -i integ_recov=tgt2 -i integ_recov=tgt3**
  - **integ_recov=tgt1** - pick the rewind and emit time of capture from a specific target (**tgt1**) only. This format can be used when there are multiple target locations and you want to recover the time of a particular target location.
  - **integ_recov=tgt1:channel1-cap-src1** - when either channel or source location name has changed in the new setup. This (**channel1-cap-src1**) also indicates the name of the old capture job (before recovery).
  - **integ_recov=tgt1:channel1-integ-tgt1** - when either channel or target location name has changed in the new setup. This (**channel1-integ-tgt1**) also indicates the name of the old integrate job (before recovery).
  - **integ_recov=tgt1:channel1-cap-src1:channel1-integ-tgt1** - when either channel or source or target location name have changed in the new setup.

	The following should be taken into consideration when executing **Integrate** after running **HVR Initialize** with **Capture Rewind**: • When **Integrate** with **/Burst** is executed then **Integrate** **/Resilient** should be defined to avoid the error triggered due to the rows that are already available (processed by a previous integrate cycle) in the target location. • When continuous **Integrate** (without **/Burst**) is executed then HVR automatically skips the rows that are already processed by a previous integrate cycle (the transaction sequence number for these rows is already available in the state tables). To forcefully **Integrate** these rows, the state tables should be cleared by selecting option **State Tables** in **HVR Initialize** dialog or specifying option **-os** in CLI. • If continuous integrate (**Integrate** without **/Burst**) is to be executed after clearing the state tables, you must define **Integrate** **/Resilient** to avoid the error triggered due to the rows that are already available (processed by a previous integrate cycle) in the target location.
-I*x*	Emit start time. This option requires **-i***time*. In HVR GUI, this option is displayed as **Emit from**. Value of *x* may be one of the following: • *time* : Emit changes from the specified moment of time. The time formats are the same as for **-i***time* option. In HVR GUI, this option is displayed as **Specific Time**. • **hvr_tx_seq**=*number* : Emit from a specific HVR transaction sequence number. The *number* can be given in a decimal or a hexadecimal form. If *number* contains decimal digits only then it is decimal. Otherwise, if it starts from prefix **0x** or contains hexadecimal digits **A**,**B**,**C**,**D**,**E** or **F** then it is treated as hexadecimal. In HVR GUI, this option is displayed as **hvr_tx_seq**. • **scn**=*number* : Emit changes from a specified **SCN**. For Oracle, this is equivalent to emit from HVR transaction sequence number where **hvr_tx_seq**=**scn*65536**. The *number* can be in a decimal or a hexadecimal form. In HVR GUI, this option is displayed as **Oracle SCN**.
-l*x*	Only affect objects for locations specified by *x*. Values of *x* may be one of the following: • */loc* : Only affect location */loc*. • */1*-*/2* : Affects all locations that fall alphabetically between */1* and */2* inclusive. • **!***/loc* : Affect all locations except */loc*. • **!***/1*-*/2* : Affects all locations except for those that fall alphabetically between */1* and */2* inclusive. Several **-l***x* instructions can be supplied together to **hvrinit**.

-o*S*	Operations limited to objects indicated by *S*. For more information about objects, see section Objects below. Value of *S* may be one of the following: • **s** : State tables in database locations, e.g. the toggle table. • **c** : Tables containing changes, e.g. burst, history and fail tables. • **t** : Database triggers/rules for tables with trigger-based capture. • **p** : Database procedures in database locations. • **r** : Router directory's transaction files. Also capture start time for log-based capture and file copy channels. • **e** : Extend enroll information for tables with log-based capture. • **E** : Recreate (replace) enroll file for all tables present in the channel. In HVR versions released between 5.3.1/5 and 5.5.0/2, enroll file is recreated only for the tables that are selected during **hvrinit**. • **l** : Supplemental logging for log-based capture tables. • **j** : Job scripts and scheduler jobs and job groups. • **f** : Files inside file location's state directory. Several **-o***S* instructions can be supplied together (e.g. **-octp**) which causes **hvrinit** to effect all object types indicated. Not specifying a **-o** option implies all objects are affected (equivalent to **-osctpreljf**).
-p*N*	Indicates that SQL for database locations should be performed using *N* sub-processes running in parallel. Output lines from each subprocess are preceded by a symbol indicating the corresponding location.This option cannot be used with option **-S**.
-r*checkpointfilepath* **Since** v5.5.5/6	Adopt most suitable retained checkpoint from checkpoint files available in *checkpointfilepath*. The *checkpointfilepath* should be the exact path for the directory containing checkpoint files. For more information about saving checkpoint files in a directory, see **Capture /CheckpointStorage**.A checkpoint file is considered not suitable when the checkpoints available in it are beyond the rewind or emit times requested for **hvrinit**. This option can be supplied more than one time with **hvrinit** to use multiple checkpoint file paths.
-S	Write SQL to **stdout** instead of applying it to database locations. This can either be used for debugging or as a way of generating a first version of an SQL include file (see action **DbObjectGeneration /IncludeSqlFile**), which can later be customized. This option is often used with options **-l***loc* **-otp**.
-t*y*	Only affect objects referring to tables specified by *y*. Value of *y* may be one of the following: • *tbl* : Only affects table *tbl*. • *t1*-*t2* : Affects all tables that fall alphabetically between *t1* and *t2* inclusive. • **!***tbl* : Affects all tables except *tbl*. • **!***t1*-*t2* : Affects all tables except for those that fall alphabetically between *t1* and *t2* inclusive. Several **-t***y* instructions can be supplied together to **hvrinit**.
-u*user*[*/pwd*]	Connect to hub database using DBMS account *user*. For some databases (e.g. SQL Server) a password must also be supplied.

Objects

The command **hvrinit** allows you to create, replace or drop objects. This section explains about every object types available for the command **hvrinit** or in the **HVR Initialize** dialog and details when it should be used. Command **hvrinit** creates objects (e.g. state tables) in the default schema for the user used for connecting to the location.

Following are the objects or advanced options for **HVR Initialize**, their meaning, and whether or not they should be enabled when running **HVR Initialize**.

State Tables

State Tables are the tables that HVR uses for processing purposes and are only relevant for database locations.

In a source database location state tables are only created for trigger-based capture scenarios. One toggle table per channel is created called **hvr_tog***chn*, and one sequence table called **hvr_seq**_*chn*. Log-based change data capture does not use state tables on the source database.

In the target database location the table names are **hvr_stbu**_*chn*_*loc*, **hvr_stin**_*chn*_*loc* and **hvr_stis**_*chn*_*loc* (with the channel name substituted for *chn* and the location name for */oc*). The state tables contain commit time and transaction information and get updated every time HVR applies transactions to this location, to ensure no transactions are lost but none are applied more than once. Generally once state tables have been created they don't have to be re-created, but if the checkbox is checked then they will be re-created, and the state data kept in the tables is lost. If it is important to keep the state of the integrate jobs in the database. Do not recreate the state tables if an error occurred such as the network connection between the hub and the destination was temporarily lost, or if a capture rewind was performed yet transactions should not be applied again to the target.

Change Tables

The option for **Change Tables** is available in GUI if a channel uses trigger-based capture on the source, if a database target uses **Integrate /OnErrorSaveFailed**, or if the action **CollisionDetect** is defined. Depending on the use case the option **Change Tables** may only source database or target database locations.

In the source database location(s) the option Change Tables creates two tables per table in the channel. Log-based change data capture does not use change tables on the source.

In a target database location if error tables exist (as a result of **Integrate /OnErrorSaveFailed**) then the error tables will be dropped with the option **Create or Replace Objects** selected (filtered based on the tables checked in the list). Error tables are only created when the first error occurs, and not during hvrinit. If the **CollisionDetect** action is defined then the history tables will be created or recreated if they already exist. If it is important to keep old error rows or history of changes for active/active environments then make sure to uncheck the option **Change Tables** when running hvrinit.

This option will also drop burst tables (tables that end with __**b**) in the target database that were created as a result of using **Integrate /Burst**. If the integration is in the middle of a burst cycle, i.e. changes were moved into the burst tables, but changes were not yet applied to the target tables, then the option Change Tables will drop data that impacts HVR's ability to recover.

Database Triggers

The option **Database Triggers** is only relevant for database capture scenarios that do have the option **Capture /TriggerBased** checked, and only affects the source database locations. Note that the triggers call database stored procedures, created by the respective option. Database triggers have to be recreated every time the definition of a table changed. Re-creating triggers is never a problem but only time-consuming if otherwise unnecessary. Filter the list of tables down to only the table(s) that require an updated trigger definition when (re)creating triggers for trigger-based capture.

If the option for **Database Triggers** is enabled but the intent is to perform log-based capture then revisit the channel definition because there may be an error in the capture action.

Trigger-based capture is supported only for certain location types. For the list of supported location types, see Trigger-based capture (action Capture **/TriggerBased**) in Capabilities.

Database Procedures

The option **Database Procedures** is relevant for the source if trigger-based capture is used, and on the target if action **Integrate /DbProc** is defined. HVR (re)generates the programming code for the stored procedures that are called by triggers for trigger-based capture, and by the integrate job for scenarios using . Database procedures must be (re)created every time a table definition changes. Filter for only the table(s) changed to speed up the creation/compilation of the stored procedures. Note that for trigger-based capture the database procedures require the state tables on the source database, and if the database is Oracle then the user for the location must have execute privileges on **DBMS_ALERT**. Without these related objects and privilege the generated programming code will fail to compile resulting in error messages.

Integrate /DbProc is supported only for certain location types. For the list of supported location types, see I ntegrate with **/DbProc** in Capabilities. The procedures rely on the table to exist (with the correct definition in sync with the table definition in the channel) in order to compile successfully. For every table three database procedures are created for insert, update, and delete, unless the table has no primary key in which case only delete and insert procedures are created (and updates are executed as a delete of the old row followed by an insert of the new row).

Transaction Files and Capture Time

For a source database location the option **Transaction Files and Capture Time** will (re)create the capture state file in **$HVR_CONFIG/router/hub/chn/loc_loc**. HVR will start capturing transactions that modify tables in the channel after the initial capture time. By default the capture time is current.

Resetting the capture time is often not desirable because any open transactions that HVR may have been tracking will be lost. If the capture is reset then typically a database refresh has to be run in order to re-synchronize tables if the capture time was reset. HVR will give a warning if the capture time is reset. Uncheck the option **Transaction Files and Capture Time** to prevent the capture time being reset.

The option **Transaction Files and Capture Time** only affects file target location.

Table Enrollment

The option **Table Enrollment** is only relevant for source database locations that use log-based capture. HVR generates an enroll file in **$HVR_CONFIG/router/hub/chn/loc_loc** listing all tables in the channel, their database object identifiers and column information in order to perform log-based change data capture. The information to generate this file is queried out of the database dictionary. Check the option Table Enrollment every time table definitions in the channel definition have changed to ensure the enroll information is up to date.

Regenerating table enrollment is always done for the entire channel and not just for one table, so if no table definitions changed then uncheck this option, especially if the channel includes many tables in which case obtaining the enroll information may take some time (also depending on the database speed).

Supplemental Logging

Supplemental logging is required to ensure table updates can be replicated correctly using SQL statements on the target database. Supplemental logging ensures that for every update to a row the database includes (at least) the primary key column data in the log. Different databases use different mechanisms and terms to enable supplemental logging.

The option **Supplemental Logging** is only relevant for source database locations when log-based capture is used. This option will enable supplemental logging on source tables as needed depending on the channel definition. HVR will implement as granular supplemental logging as possible but many options in the channel, as well as the features of the database, determine whether full supplemental logging on all columns is required, or only for a subset of the columns e.g. the primary/unique key.

Validating whether the correct supplemental logging is in place can take a significant amount of time – proportional to the number of tables in the channel, so if no tables have been added to the channel and no tables where dropped and recreated in the source database then uncheck this option when running hvrinit. If only few tables were added and others already had supplemental logging added then consider filtering the list of tables to only the tables that still need supplemental logging to speed up running hvrinit.

For Oracle, the option **Supplemental Logging** in conjunction with **Drop Objects** will not drop the supplemental logging since HVR does not know if other software relies on the supplemental logging HVR may or may not have created.

Scripts and Jobs

The option **Scripts and Jobs** registers the jobs under the **HVR Scheduler** (and in the respective database tables in the HVR hub database). Scripts and jobs always have to be recreated after a change is made to the channel.

- For a source database location, a job file is generated in **$HVR_CONFIG/job/hub/chn**.
- For a target database location, a **.cache** file is generated in **$HVR_CONFIG/router/hub/chn /catalog**.

File Location State

The option **File Location State** (option **-of**) resets the directory **_hvr_state** in a file location. This option is only relevant for file locations.

Capture Rewind and Emit Time

HVR uses the following concepts for managing the capture start time:

- Capture Rewind
- Emit Time

Capture Rewind

The capture rewind option instructs HVR to capture changes starting from a specific time in the past, rather than the moment the **Hvrinit** was run. Refer to option -i to see all the values available for managing the capture rewind time.

Use the **Capture Rewind** option to go back to an earlier point in time if the table definitions were identical to the current table definitions (matching the definitions in the channel) and supplemental logging was enabled on the tables at the earlier point in time. Old transaction log files must still be available to rewind back to the earlier point in time. **Capture Rewind** is not available for trigger-based capture.

Resetting the capture time is often not desirable because any open transactions that HVR may have been tracking will be lost. If the capture is reset, then a database refresh has to be run in order to re-synchronize tables. HVR will give a warning if the capture time is reset. Unselect option **Transaction Files and Capture Time** to prevent the capture time from being reset.

Emit Time

Emit time refers to the point at which changes will be sent from the capture location to the integrate location. The **Emit Time** may differ from the **Capture Rewind** time for a system with long-running transactions. HVR will start capturing changes to tables in a channel only from the capture time forward. However, in order to fully capture long-running transactions in the system, you need to start capture earlier and emit only from some point forward. ERP systems in an Oracle database often have long-running transactions, but long-running transactions are not common in many other databases.

For example, if there is a long transaction that lasts 1 hour from open to commit. To capture this entire transaction, HVR has to start reading database logs from at least 1 hour ago. Suppose there are other small transactions that got committed in the meantime, which do not need to be captured. Then the **Emit Time** would be near the end of this long transaction. So, the **Capture Rewind** time controls where HVR starts reading data from (must be before the start of any transactions we are interested in) and the **Emit Time** controls the end of the transaction. If a transaction (tx) file is committed before the **Emit Time**, HVR ignores it, if it is committed after the **Emit Time**, HVR captures it and writes transaction files.

Capture Rewind Scenarios

There are several scenarios when the **Capture Rewind** option is selected as part of replication initialization and a “reset” (i.e. **Hvrrefresh**) is not required.

For example, if a source database is upgraded from one version to another while the application is not running. Upgrading a database often results in a lot of transaction log changes, and regardless of whether HVR can read those changes without any issues, it may just be cleaner to suspend the capture during the upgrade. It is also possible that the database will be restarted as part of the upgrade, probably more than once. Then, after the upgrade, you will perform the capture rewind to a current point (or to the point in time after the upgrade, but before starting the application) to skip over the transaction log changes during the upgrade. In such a scenario, you would know that you have not missed any changes to the application.

Another example would be an application upgrade with downtime, during which there will also be no changes to the tables to be replicated. In some cases, there are table changes (DML and/or DDL) made during the application upgrade that may or may not need to be reflected as part of the replication.

Options Tab in HVR Initialize

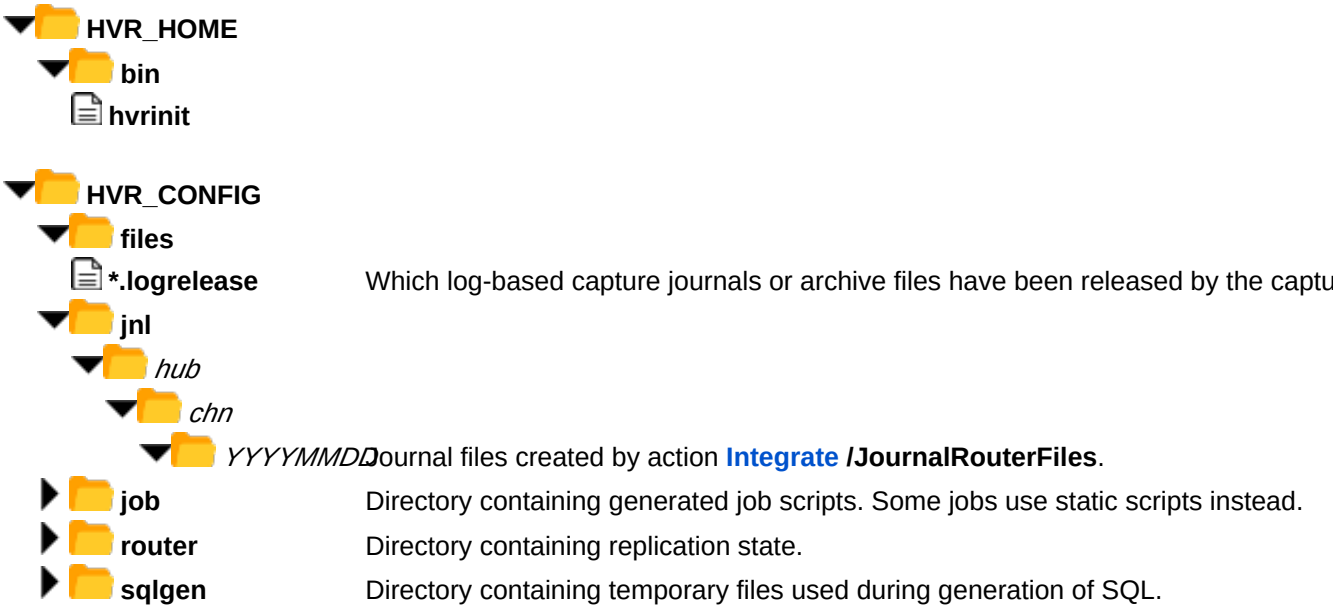
HVR Initialize Objects and Capture Rewind options are displayed under this tab. Depending on the actions defined in the channel only few of the options are applicable/enabled. HVR automatically disables options that are not relevant to the channel based on the channel definition. For example, the option **Datab ase Triggers** is disabled for channels that use log-based capture.

Locations Tab in HVR Initialize

By default all locations are checked, so by default the selected advanced options apply to all locations. Many of the advanced options can be re-run without an affect on the replication setup but hvrinit will finish faster if it only runs for a limited set of locations.

The recommendation is to check only the location(s) that are affected by a change to the channel. E.g. for a change to the group representing the capture side, only select the source location(s). For a change only affecting the integrate side, only check the target location(s). If a new location was added then only run hvrinit for the new location.

Files



See Also

Commands Hvrproxy, Hvrrouterview and Hvrscheduler.