Hvrscheduler

Contents

Name

hvrscheduler - HVR Scheduler server.

Synopsis

hvrscheduler [*-options*] *hubdb*

Description

The HVR Scheduler is a process which runs jobs defined in the catalog table **HVR_JOB**. This catalog table can be found in the hub database.

These jobs are generated by commands **Hvrinit**, **Hvrrefresh** and **Hvrcompare**. After they have been generated these jobs can be controlled by attributes defined by the jobs themselves and on the job groups to which they belong. These attributes control when the jobs get scheduled.

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.

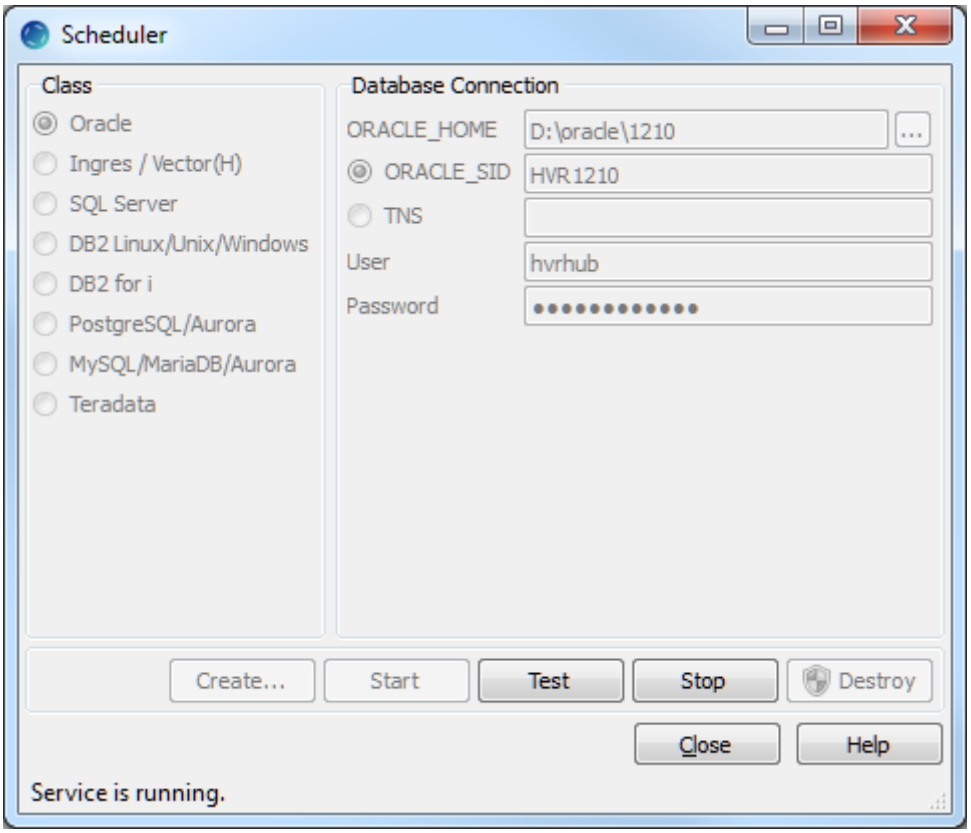
On Unix, the HVR Scheduler runs as a daemon. It can be started and stopped within the HVR GUI. Alternatively, on the Unix command line, it can be started using command **hvrscheduler** *hubdb* (no options) and stopped using **hvrscheduler -k**.

On Windows, the HVR Scheduler runs as a system service. It can be started and stopped within the HVR GUI. Alternatively, on the Windows command line, it can be created using command **hvrscheduler -ac**, started with **hvrscheduler -as** and stopped with **hvrscheduler -ah**.

Internally the HVR Scheduler uses a concept of 'Job Space', a two-dimensional area containing jobs and job groups. A job group may contain jobs and other job groups. In Job Space, jobs are represented as points (defined by X and Y coordinates) and job groups are represented as boxes (defined by four coordinates minimum X, maximum X, minimum Y and maximum Y). All jobs and job groups are contained within the largest job group, which is called **system**.

Options

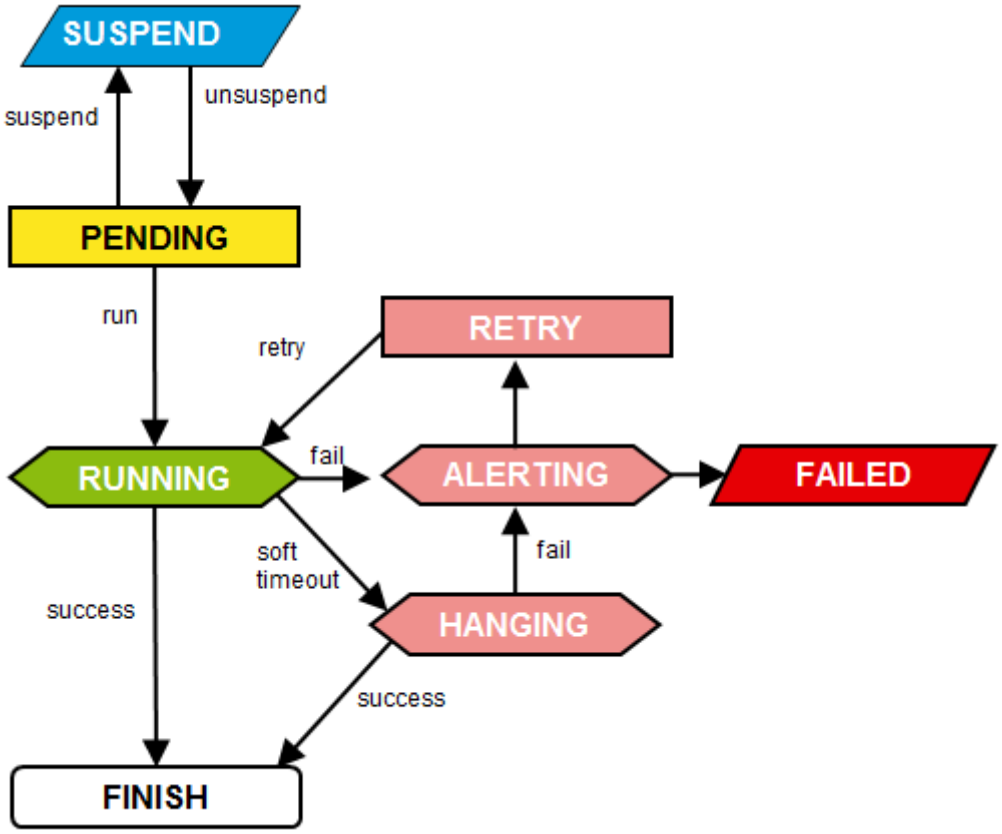
This section describes the options available for command **hvrscheduler**.



Parameter	Description
-a*x* **Windows**	Administration operations for the HVR Scheduler Microsoft Windows system service. Allowed values of *x* are: • **c** : Create the HVR Scheduler service and configured it to start automatically at system reboot. The service will run under the default system unless **-P** option is given. • **s** : Start the HVR Scheduler service. • **h** : Halt (stop) the system service. • **d** : Destroy the system service. Several **-a***x* operations can be supplied together, e.g. **-acs** (create and start) and **-ahd** (halt and destroy). Operations **-as** and **-ah** can also be performed from the window **Settings ControlPanel Services** of Windows.
-c*clus*|*clusgrp* **Windows**	Enroll the Scheduler Service in a Windows cluster named *clus* in the cluster group *clusgrp*. Once the service is enrolled in the cluster it should only be stopped and started with the Windows cluster dialogs instead of the service being stopped and started directly (in the Windows Services dialog or with options **-as** or **-ah**). In Windows failover clusters *clsgrp* is the network name of the item under **Services and Applications**. The group chosen should also contain the DBMS service for the hub database and the shared storage for **HVR_CONFIG**. The service needs to be created (with option **-ac**) on each node in the cluster. If this option is used to create the scheduler service in a cluster group, then it should also be added to option **-sched_option** of command **Hvrmaint**. This service will act as a 'Generic Service' resource within the cluster. This option must be used with option **-a**.
-E*n*=*v*	Set environment variable *n* to value *v* for this process and its children.
-F	Force start the **HVR Scheduler** process. This overrides certain checks that the scheduler does before starting. This is an internal option used by HVR.

-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	Interactive invocation. **HVR Scheduler** does not detach itself from the terminal and job output is written to **stdout** and **stderr** as well as to the regular logfiles.
-k Unix & Linux	Kill currently running **HVR Scheduler** process and any jobs which it may be running at that moment. When this option is used, it contacts the **HVR Scheduler** process to instruct it to terminate by itself. If this does not happen, it will kill the **HVR Scheduler** process using its process id (PID).
-K Unix & Linux	Kill immediately the currently running **HVR Scheduler** process and any jobs which it may be running at that moment. This is a variant of option **-k** except it skips the initial step of contacting the **HVR Scheduler** process and instructing it to terminate by itself.
-P*pwd* Windows	Configure the **HVR Scheduler** system service to run under the current login account using password *pwd*, instead of under the default system login account. May only be supplied with option **-ac**. Empty passwords are not allowed. The password is stored (hidden) within the Microsoft Windows OS and must be re-entered if passwords change.
-q*secs* Unix & Linux	Kill the **HVR Scheduler** process and any jobs which it may be running at that moment. This option allows *secs* seconds grace time before terminating the **HVR Scheduler** process. The **default** is 60 seconds. This parameter can also be passed from **hvrmaint** using the **-quiesce_grace** option.
-s*lbl*	Add label */lbl* to HVR's internal child co-processes. The scheduler uses two co-processes at runtime; one to make SQL changes to the hub database (**-swork**), and the other for listening for database events (**-slisten**).
-t*secs*	Connection timeout after *secs* seconds to the old **HVR Scheduler** process. The **default** is 10 seconds.
-u*user*[*/pwd*]	Connect to hub database using DBMS account *user*. For some databases (e.g. SQL Server) a password *pwd* must also be supplied.

Job States



The HVR Scheduler schedules jobs. Each job performs a certain task. At any moment a job is in a certain state. For instance, when a job is waiting to be run, it is in state **PENDING**; when a job is running, it is in state **RUNNING**.

Jobs can be either acyclic or cyclic. Acyclic jobs will only run once, whereas cyclic jobs will rerun repeatedly. When a cyclic job runs, it goes from state **PENDING** to **RUNNING** and then back to state **PEND ING**. In this state it waits to receive a signal (trigger) in order to run again. When an acyclic job runs, it goes from state **PENDING** to **RUNNING** and then disappears.

If for some reason a job fails to run successfully the scheduler will change its state first to **ALERTING**, then **RETRY** and will eventually run again. If a job stays in state **RUNNING** for too long it may be marked with state **HANGING**; if it finishes successfully it will just become **PENDING**.

Output Redirection

Each message written by an HVR job is redirected by the scheduling server to multiple logfiles. This means that one logfiles exists with all output from a job (both its **stdout** and **stderr**). But another file has the **stderr** from all jobs in the channel.

Scheduler Attributes

Scheduler attributes are a component which is used to internally communicate (at the moment that HVR Initialize is run) the definition of features such as **Scheduling /CaptureStartTimes** to the run-time system. They are exposed in the HVR User Interface to allow the verification that these **Scheduling** actions have been propagated to the run-time system.

These scheduler attributes will be redesigned in a future HVR version. It is recommended not to change scheduler attributes that HVR generates automatically or not to create new ones.

Attribute	Arg1	Arg2	Description

quota_run	*n*		Maximum number of jobs which can be in **RUNNING** or **HANGING** state in job group. So if attribute **quota_run 2** is added to job groups **CHN1**, **CHN2** and **quota_run 3** is added to job group **SYSTEM**, then only two jobs can run for each channel (**CHN1** and **CHN2**) and only three jobs can run in the whole system.
quota_children	*n*		Maximum number of child processes associated with jobs in job group, including running jobs, but excluding the scheduler's own child processes.
quota_speed	*q*	*secs*	Limit the speed with which the scheduler starts jobs to no more than *q* job executions inside *secs* seconds. For example, **quota_speed 20 1** means that if there are lots of job ready to be started, then the scheduler will only start 20 jobs per second.
trig_delay	*secs*		Trigger cyclic job *secs* seconds after it last finished running (column **job_last_run_end** of **HVR_JOB**). If a cyclic job is not affected by any **trig_delay** attribute, it will remain **PENDING** indefinitely.
trig_crono	*crono*		Trigger job at *crono* moment. For the format of crono see **Scheduling /CaptureStartTimes**. After applying attribute **trig_crono**, the job needs to be in a **PENDING** state for the **HVR Scheduler** to trigger the job.
trig_at	*time*		Trigger job at specific (*time*) moment. Valid formats are *YYYY-MM-DD [HH:MM:SS]* (in local time) or *YYYY-MM-DDTHH:MM:SS+TZD* or *YYYY-MM-DDTHH:MM:SSZ* or **today** or **now***[±SECS]* or an integer (seconds since **1970-01-01 00:00:00 UTC**).
retry_max	*n*		Allow *n* retries for an unsuccessful job. If *n* is zero no retry is allowed. Jobs unaffected by this attribute are not retried: on error they become **FAILED** directly.
retry_delay	*isecs*	*fsecs*	Initially retry job after *isecs* and double this delay for each unsuccessful retry until *fsecs* is reached. The default value for *isecs* is **60** seconds and *fsecs* is **3600** seconds.
timeo_soft	*secs*		After job has been in **RUNNING** state for *secs* seconds, write time-out message and change its job state to **HANGING**. If *secs* is zero then no time-out will occur.
timeo_hard	*secs*		Terminates the job in **RUNNING** or **HANGING** state after it has run *secs* seconds. If *secs* is zero then no time-out will occur.
set	*name*	*val*	Set variable name to value *val* in job's runtime environment.

Environment Variables

Variable Name	Description

HVR_ITO_LOG	Causes the HVR Scheduler to write a copy of each critical error tot the file named in the variable's value. This can be used to ensure all HVR error messages from different hub databases on a single machine can be seen by scanning a single file. Long messages are not wrapped over many lines with a backslash '****', but instead are written on a single line which is truncated to 1024 characters. Each line is prefixed with "**HVR_ITO_AIC** *hubnode:hubdb locnode*", although **HVR** is used instead of **$HVR_ITO_AIC** if that variable is not set.
HVR_PUBLIC_PORT	Instructs the HVR Scheduler to listen on an additional (public) TCP/IP port number.

Examples

In Unix, start **HVR Scheduler** as a Unix daemon.

```
hvrscheduler hubdb
```

In Windows, create and start **HVR Scheduler** as a Windows Service.

```
hvrscheduler -acs hubdb
```

When starting the **HVR Scheduler** it is important that a database password is not exposed to other users. This can be encrypted using command **hvrcrypt**.

Files

- **HVR_HOME**
  - **bin**
    - **hvralert** — Perl script used by scheduler to decide if jobs should be retried.
  - **lib**
    - **retriable.pat** — Patterns indicating which errors can be handled by retrying a job.
- **HVR_CONFIG**
  - **files**
    - **sched***db_node*.**pid** — Process-id file.
    - **sched***db*.**host** — Current node running scheduler.
  - **log**
    - **hubdb**
      - *job*.**out** — All messages for job *jobname*.
      - *job*.**err** — Only error messages for job *jobname*.
      - *chn*.**out** — All messages for channel *chn*.
      - *chn*.**err** — Only error messages for channel *chn*.
      - **hvr.out** — All messages for all jobs.
      - **hvr.err** — Only error messages for all jobs.
      - **hvr.ctrl** — Log file for actions from control sessions.
- **HVR_TMP** — Working directory for executing jobs.

See Also

Commands Hvrcrypt, Hvrsuspend, Hvrstart.