insightsoftware

Hubble Desktop and Web

Accelerator User Guide

Version 4.3

January 2025

Contents

Document Information

Notices

Copyright

Disclaimer

This guide is designed to help you to use the Hubble applications effectively and efficiently. All data shown in graphics are provided as examples only. The example companies and calculations herein are fictitious. No association with any real company or organization is intended or should be inferred.

Version History

Date	Revision	Software Version	Comments
7th January, 2025	1.0	4.3	Initial issue for 4.3.

Customer Support

For more information regarding our products, please contact us at https://insightsoftware.com/hubble/.

For product support including Training, Documentation and Customer Support, visit our Hubble Community at https://help.insightsoftware.com/.

Conventions Used

The icons below are used in Hubble documentation to indicate type of user, experience, deployment and the ERP system applicable to the document.

ICON	USER TYPE	ICON	EXPERIENCE/ DEPLOYMENT	ICON	ERP TYPE
	Viewer Viewer users can consume content and contribute to planning.		Desktop Experience		JD Edwards
	Power Power users can consume or create content, administer and contribute to planning.		Web Experience		Oracle
	Designer Designer users can consume, create and extend content, as well as build, administer and contribute to planning.		On-Premise Deployment (for Web Experience)		
	Budgeting/Planning Users with this capability can do real-time budgeting and forecasting.		Cloud Deployment (for Web Experience)		
	Administrator Administrator users administer the Hubble Product.				

Recommended Hardware Configuration

Scalability Guidelines Based On Customer Data

Based on testing and production results from several customers, Table 1 offers guidelines for expected user and/or data densities. Results vary based on many variables and doing your own testing will determine your specific scalability.

Table 1: Hardware/VM Usage Guidelines

	Performance Variables	Minimum	Recommended	Advanced
1	Data volumes in Vector - Low (~20GB)	X	X	
2	Data volumes in Vector - Medium (~150GB)		X	
3	Data volume in Vector- High (~500GB)			X
4	Data complexity [1] - Low (<100k transactions per day)	X	X	
5	Data complexity [1] - Medium (100k - <1M transactions per day)		X	
6	Data complexity [1] - High (>1M transactions per day)			X
7	Anticipated User Count [2]	20-50	30-100	100-1000+

[1] Complexity in this context is volume and frequency of change via Change Data Capture (CDC).

[2] Precise user counts will vary subject to the complexity of application usage, workspaces, reports, etc.

Tip: We recommend early discussions with the Hubble Support on appropriate hardware configuration for significantly higher user counts (and/or multiple data sources).

Analysis Summary

The combination of Data Volume and Data Complexity require significant CPU resources, so investing in higher CPU speeds and more cores will pay off in both performance and scalability. In Table 1 above, the combination of #1 and #5 may require use of the Advanced specifications as shown in the last column.

Hubble Accelerator Host Server Hardware

Table 2: Accelerator Hardware Host Server Guidelines

Aspect	Minimum	Recommended	Advanced [1]
CPU (or greater)	Dual Intel® Xeon® Processor E5-2640 v3 (20M Cache, 2.60 GHz) [16 cores total]	Dual Intel® Xeon® Processor E5-2690 v3 (30M Cache, 2.60 GHz) [24 cores total]	Dual Intel® Xeon® Processor E5-2699 v4 (55M Cache, 2.20 GHz) [44 cores total]

Aspect	Minimum	Recommended	Advanced [1]
Memory	>192 GB	256 GB	256-512 GB
Networking [2]	1 Gbps	10 Gbps	10 Gbps
Storage [3]	Sufficient for hypervisor plus hosted systems. High speed (1+ GB/sec read) SSDs[4]recommended.		

[1] Use of dual 4th Generation Intel® Xeon® Scalable Processors may be appropriate in some cases (beyond 250 concurrent users with multiple complex data sources, for example). Please discuss exact requirements with Hubble Support at early stages of your planning

[2] 1 GB/sec equates to read IOPS of 266,000 or greater.

[3]The primary storage for the Vector database is disk storage. Your storage solution must satisfy performance and space requirements. To achieve good consistent performance, you should choose smaller rather than bigger disks.
For example, choose 146 GB disks over 500 GB (or larger) disks. Faster spinning disks at 15k RPM have higher throughput rates than slower 10k RPM or 7.2k RPM disks. In an ideal case, a single spinning 15k RPM disk can sustain up to 150 MB/s data transfer.

[4]Vector is optimized to work with both memory and disk-resident datasets, allowing it to efficiently process large amounts of data (hundreds of gigabytes). Vector can process data at more than 1.5 GB/sec per CPU core. To achieve this rate, the CPU cores must be fed at a rate fast enough to keep them busy. Consider making SSD technology a viable storage consideration for your system. It is recommended to choose SSDs for temporary database storage to improve performance for spill-to-disk operations i.e. Vector work area (II_WORK).

Hubble Accelerator Virtual Machine

Note: For optimal performance, it is recommended to run the Hubble Accelerator in a Virtual Machine on dedicated hardware. It should not be hosted on the same physical device as other Virtual Machines or business applications that could impact system resources.

Table 3: Hubble Accelerator VM Guidelines

Aspect	Minimum	Recommended	Advanced
Memory [1]	128 GB	192 GB+	192-468 GB
vCPUs	32 cores	48 cores	88 cores
Disk Space [2]	Data [2] - 500 GB	Data [2] - 1 TB	Data [2] - 1-2 TB

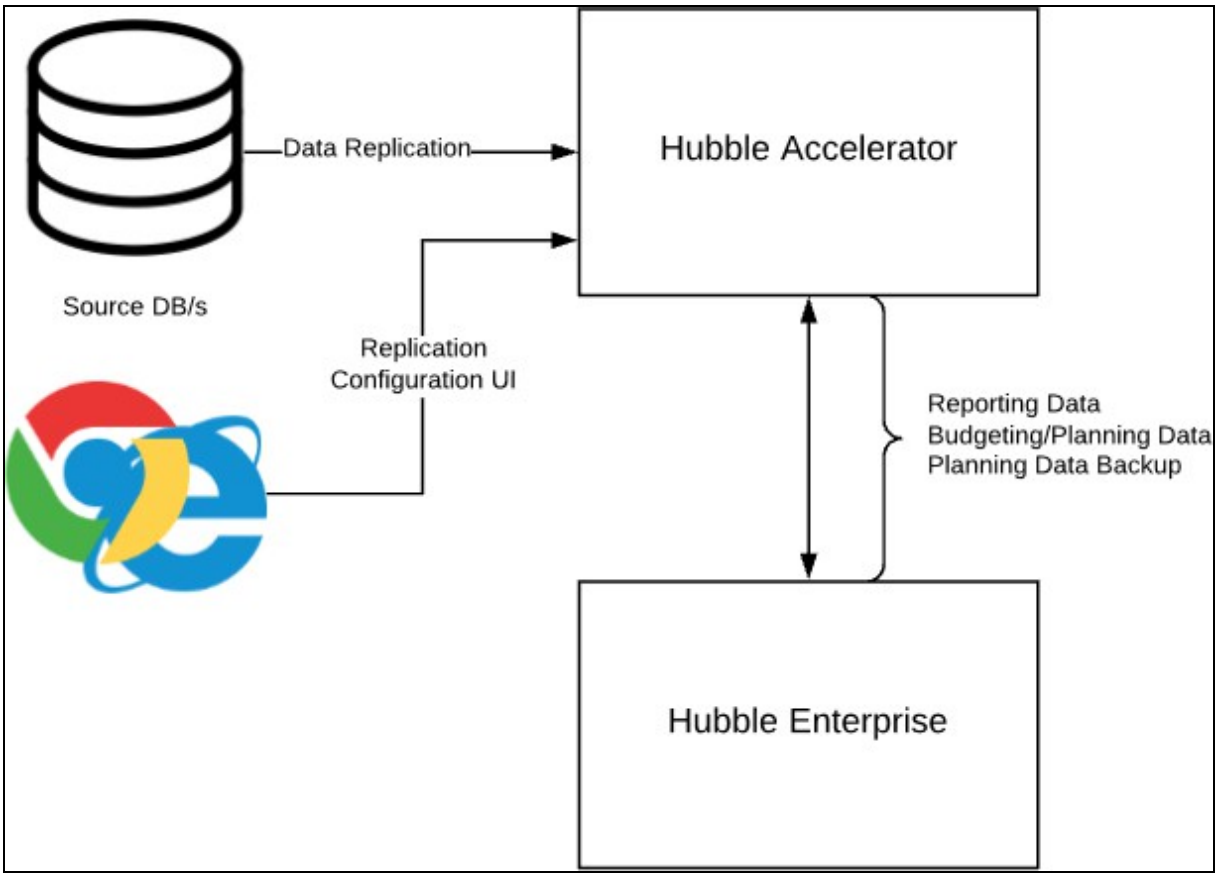
[1]This is the amount of RAM available to the applications and is based on a single replication data source. Additional RAM may be required for particularly complex or big data environments. RAM size may have to be increased post-implementation.

[2] Storage requirements will vary considerably by customer, depending upon the quantity of source ERP data and how much needs to be replicated to the Accelerator to service your reporting and analytics needs. For guidance, the typical compression ratio between source data in the ERP database and the accelerator is 4:3.

Architecture

System Architecture

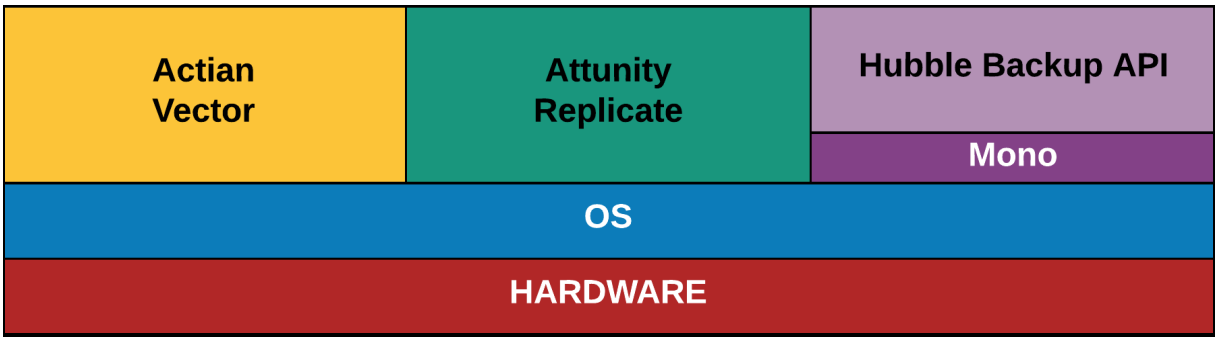
Below is a diagram with the standard System Architecture of a Hubble Enterprise Deployment.



Components

Hubble accelerator installs:

- Actian Vector
- Attunity Replicate
- Hubble Backup API



Network Access

Below are defined all the ports a firewall needs to be configured with to ensure normal function of Hubble Accelerator as part of a Hubble deployment.

Port	Protocol	Service	Software	Sources
3552	TCP	Replication Admin UI	Attunity Replicate	Replication Admin PC
27832	TCP	Net server IIGCC (Communication)	Actian Vector	Hubble desktop users on company network, web server
27839	TCP	Data Access Server	Actian Vector	Hubble desktop users on company network, web server
3550	TCP	Replication	Attunity Replicate	Qlik front end running on the accelerator
5000	TCP	Hubble Backup API	hubble-applianceapi	Admin tool backup budgeting data
50001	TCP	Hubble Alert Services	hubble-alerts	Application server, Web Servers

Install Hubble Accelerator

This section describes how to prepare your system for Hubble Accelerator and how to install it.

Installation Prerequisites

This section describes how to prepare your system to use Hubble Accelerator. The requirements differ according to the platform on which you want to install Hubble Accelerator.

Software Requirements

Hubble Accelerator is distributed in RPM format and can be installed on the following Linux platforms

- Red Hat Enterprise Linux (64-bit) 7.x
- CentOS 7.x

Dependencies

All dependencies and its dependencies are required. If an offline installation is required, please ensure all dependencies are installed before installing Hubble Accelerator rpm.

- libaio
- perl
- libX11
- libXext
- libXi
- libXrender
- libXtst
- alsa-lib
- perl
- perl-Data-Dumper
- unixODBC-utf16 (For version 4.1.x or higher - shipped with the RPM package)
- initscripts
- atk
- cairo
- gdk-pixbuf2

- gtk2
- pango
- sudo
- which
- mono-complete-6.12.0.107-0.xamarin.9.epel8 (for version 4.2.x or higher)
- bc
- e2fsprogs
- openssl

To install **mono-core-6.12.0.107-0.xamarin.9.epel7.x86_64** on CentOS/RHEL 8:

- Refer to the detailed instructions for installing Mono on CentOS/RHEL 8 available at https://www.mono-project.com/download/stable/#download-lin-centos
- For additional information, you can check the release notes for Mono 6.12.0 at https://www.mono-project.com/docs/about-mono/releases/6.12.0/.

If some dependencies for **mono-complete-6.12.0.107-0.xamarin.9.epel7** have been moved from the **centos8-stable** repository, you can find them at this URL. You can manually download and install them using the following commands:

- Download the package:

```
wget <url>
```

- Install the package:

```
yum localinstall <package.rpm>
```

Uninstallation

To uninstall **unixODBC-utf16** for an older version of Accelerator (for example, Version 3.x), you can use the following command:

```
yum remove -y unixODBC-utf16
```

To uninstall **unixODBC** for version 4.2.x or higher, you can use the following command:

```
yum remove -y unixODBC
```

To uninstall Mono for version 4.2.x or higher, you can use the following command:

```
As root user
# yum list mono-complete --showduplicate

Remove older versions
# yum remove -y mono

# cd /etc/yum.repos.d/
# ls -la centos7-stable.repo
# rm centos7-stable.repo

Download centos8-stable repo
# wget https://download.mono-project.com/repo/centos8-stable.re
# yum install mono-complete
```

Supported Replication Source Endpoints

Oracle

For supported Oracle DB versions and configurations, refer to Qlik Replicate's documentation.

Oracle Instant Client version 12.1.0.2.0-1.x86_64 will be installed with Hubble Accelerator installation.

Microsoft SQL Server

For supported Microsoft SQL Server DB versions and configurations, refer to Qlik Replicate's documentation.

Microsoft ODBC Driver 18.1 for Windows will be installed on the Qlik Replicate Server machine with Hubble Accelerator installation (for version 4.2.x or higher).

Important: Installing MSODBC 18.1 will uninstall previous versions such as 13.1 and 17.1.

TLS 1.2 prerequisites

If your environment includes:

- Microsoft SQL Server 2014
- Microsoft ODBC Driver 18 installed on Red Hat 8.1 or later

To enable TLS 1.2, you must install a specific Service Pack on the Windows machine where Microsoft SQL Server is installed. For more details, visit https://support.microsoft.com/en-us/help/3135244/tls-1-2-support-for-microsoftsql-server.

IBM DB2 for z/OS

For supported IBM DB2 for iSeries versions and configurations, refer to Qlik Replicate's documentation.

IBM i Access ODBC 1.1.0.27-1.0 will be installed with Hubble Accelerator installation (for version 4.2.x or higher).

Recommended Host Configuration

Provide a separate disk to hold the Vector database and Qlik replication Data. For example, `/etc/fstab`:

```
LABEL=hubble    /opt    auto    defaults    00
```

Users and Groups

Hubble Accelerator installer will create users and groups in Linux.

Users:

- **attunity:** Owner of all Attunity related files and directories. This user runs Attunity Replicate on the Accelerator.
- **actian:** Owner of all Actian related files and directories. This user runs Actian Vector and its related executable on the Accelerator.
- **hubble-applianceapi:** Owner of Hubble Appliance API related files and directories.

Groups:

- **actian**
  - members: actian
- **attunity**
  - members: attunity
- **hubble**
  - members: hubble-applianceapi

Installation Steps

1. Install the local Hubble Accelerator rpm (replace **4.x.x-x** with the actual version of the rpm, example: **4.2.1-29**):

   ```
   yum localinstall hubble-accelerator-4.2.1-29.el8.x86_64.rpm
   ```

This step will create the Hubble Accelerator directories and copy the installation packages for all the Accelerator components.

2. Comment `${ACCELERATOR_DIR}/post-install.sh` script entry to edit Vector DB parameter settings (Step 4 and 5).

```
cat /opt/insightsoftware/hubble-accelerator/4.2.1/install.sh
#!/bin/sh
set -e

# Accelerator install
SCRIPT_DIR=$(dirname "$0")
ACCELERATOR_PROFILE=/etc/profile.d/accelerator.sh
ACCELERATOR_SOURCE=${SCRIPT_DIR}/.env
. ${ACCELERATOR_SOURCE}
. ${SCRIPT_DIR}/library.sh

echo "echo \"Hubble Accelerator: '${ACCELERATOR_VERSION}'\";" >
${ACCELERATOR_PROFILE}

# Install Vector
sh ${ACCELERATOR_DIR}/vector/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"

# Install Vector patch
sh ${ACCELERATOR_DIR}/vector/install-patch.sh -p
"${ACCELERATOR_PROFILE}" -d "${ACCELERATOR_DIR}"

# Install Vector
sh ${ACCELERATOR_DIR}/vector/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"

# Install Attunity Replication
sh ${ACCELERATOR_DIR}/attunity/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"
```

```
# Install Oracle ODBC
sh ${ACCELERATOR_DIR}/oracle/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"

# Install MSODBC
sh ${ACCELERATOR_DIR}/msodbc/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"

# Install IACCESSODBC
sh ${ACCELERATOR_DIR}/iaccessodbc/install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"

# Install Hubble Appliance API
sh ${ACCELERATOR_DIR}/hubble-applianceapi/install.sh -p
"${ACCELERATOR_PROFILE}" -d "${ACCELERATOR_DIR}"

# ${ACCELERATOR_DIR}/post-install.sh -p "${ACCELERATOR_
PROFILE}" -d "${ACCELERATOR_DIR}"
```

3. Run the Hubble Accelerator installer that will install all the rpms for all the components, and configure them:

   Note: Replace **4.x.x** with the actual version of the rpm, for example, **4.2.1**.

   ```
   /opt/insightsoftware/hubble-accelerator/3.x.x/install.sh
   ```

4. For Accelerator build **4.x.x** with Vector version **6.x.x** or later, delimited identifiers and case sensitivity database parameter settings need to be catered before creating vector database and users (`Post-install.sh script`) during Accelerator installation. Qlik Replicate will create tables and columns with upper or lower cases as per source database metadata and will cause issue with Hubble Application accessing tables/columns with delimited identifiers and upper cases. Make the following changes to update the `config.dat` database parameter entry (**localhost.createdb.delim_id_case : lower**) before creating the default database 'db' and users.

   Note: Hubble supports table and column names in lowercase.

To set `ii.localhost.createdb.delim_id_case` to lowercase in `config.dat`:

a. Go to the directory:

```
cd $II_SYSTEM/ingres/files/
```

b. Open `config.dat`:

```
vi config.dat
```

c. Find and change the line to:

```
ii.localhost.createdb.delim_id_case : lower
```

d. Save and exit:

```
:wq!
```

5. For Accelerator build **4.3.x** with Vector version **6.3.x** or later, Actian has provided the flexibility for switching back to the old behavior of not padding spaces to strings using CHAR / NCHAR functions. When CHAR and NCHAR types are used with x100 tables, the correct behavior is to pad these datatypes with spaces to the explicit and/or implicit length of the datatype. This parameter x100_nopad_char_compat adds a backward compatibility configuration option to allow the user to fall back to the old behavior (Vector version **5.x**) where these datatypes were not padded.

   To set `ii.localhost.dbms.*.x100_nopad_char_compat` to ON in `config.dat`:

   a. Go to the directory:

   ```
   cd $II_SYSTEM/ingres/files/
   ```

   b. Open `config.dat`:

   ```
   vi config.dat
   ```

   c. Find and change the line to:

   ```
   ii.localhost.dbms.*.x100_nopad_char_compat: ON
   ```

   d. Save and exit:

```
:wq!
```

In Vector version **6.3,** the default configuration for string truncation is set to "fail," whereas in Vector version **5.0**, it was "ignore."

Update the configuration parameter as follows:

```
ii.<host>.config.string_truncation: ignore
```

6. After updating the `config.dat` file, restart the vector instance and run the post-installation script to implement the non-default parameter. Use the following commands as the actian or root user:

   Restart the Vector instance (in case of upgrading an Accelerator (for implementing non-default Db parameter) after updating `config.dat` file using below commands:

   ```
   systemctl restart actian-vectorVW
   ```

   or

   Restart the vVctor instance and run post installation script (in case of fresh installation of an Accelerator for implementing non-default parameter) after updating config.dat file using below command:

   ```
   systemctl restart actian-vectorVW Ex:
   ./opt/insightsoftware/hubble-accelerator/x.x.x/post-install.sh
   -p /etc/profile.d/accelerator.sh -d
   /opt/insightsoftware/hubble-accelerator/x.x.x/
   ```

7. Reboot the machine.

8. In case of upgrade, for version **3.3.x** or higher, the current vector database 'db' needs to be upgraded:

   ```
   upgradedb db
   ```

9. Test if you can access the Vector database:

   ```
   sql db
   ```

10. Confirm that replication is installed.

```
repctl version
```

11. The password for the replication user, and the Hubble user for the database will be in:

    ```
    cat /opt/insightsoftware/hubble-accelerator/.vector
    ```

    Feel free to remove the password from that file for security reasons, but leave the file empty.

12. The password for the replication user, in the replication console can be found in:

    ```
    cat /opt/insightsoftware/hubble-accelerator/.attunity
    ```

    Feel free to remove the password from that file for security reasons, but leave the file empty.

13. Enable and start the Hubble Appliance API service:

    ```
    systemctl enable hubble-applianceapi
    ```

    ```
    systemctl start hubble-applianceapi
    ```

Uninstall Hubble Accelerator

1. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

2. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

3. Stop the Hubble Appliance API Service:

```
systemctl stop hubble-applianceapi
```

4. Run the uninstall script, which will uninstall Vector, Replication and the ODBC drivers used to connect to replication sources

> **Note:** Replace 4.x.x with the actual version you wish to uninstall.

```
/opt/insightsoftware/hubble-accelerator/4.2.x/uninstall.sh
```

5. Uninstall the Hubble Accelerator RPM package.

> **Note:** Replace 4.x.x with the actual version you wish to uninstall.

```
yum remove hubble-accelerator-4.2.1-29.el8.x86_64
```

> **Note:** Uninstalling will not affect the data for Vector and Replication, it will remain intact.

Upgrade Hubble Accelerator

There are two main ways to upgrade Hubble Accelerator:

- Upgrading/Migrating from Accelerator Versions Prior to 3.x
- Upgrading from a 4.x Version to another 4.x Version

Upgrading/Migrating from Accelerator Versions Prior to 3.x

1. On the old **Hubble Accelerator 2.x** machine:
   a. Stop the Vector Service:

   ```
   /etc/init.d/S60ingresVW stop
   ```

   b. Stop the Replication Service:

   ```
   /etc/init.d/S65areplicate stop
   ```

   c. Back up the current data disk (disk mounted on `/mnt/ybdata`).

   d. Detach the data disk.

2. On the new **Hubble Accelerator 4.x** machine:
   a. Install the Hubble Accelerator 4.x (see Install Hubble Accelerator).

   b. Attach the data disk from the old Hubble Accelerator to the new machine:

   i. Run the following command::

   ```
   blkid
   ```

   You should see output similar to this. Locate the device with **LABEL="ybdata"** as it contains the old data:

   ```
   /dev/xvda2: UUID="4a1c93d9-eb47-4f96-9f3d-920e52dc8cca"
   TYPE="xfs" PARTUUID="078b129f-9e3b-4665-8871-232657ae682c"
   ```

```
/dev/xvdb1: LABEL="ybconfig" UUID="213187cf-5d80-4623-b4fc-53fa5153cd35" TYPE="ext2" PARTUUID="897ae9ad-83e2-4e7c-8dd8-686cd0d36837"
```

```
/dev/xvdb2: LABEL="ybdata" UUID="9a7fc249-8fd7-40ed-84a6-6789c8346f78" TYPE="ext3" PARTUUID="4c5d92db-4cf2-40f6-b0f1-c28d1f67652d"
```

```
/dev/xvda1: PARTUUID="e7184259-29aa-42a5-b24d-1143dc10e5d8"
```

ii. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

iii. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

iv. Edit /etc/fstab

```
vi /etc/fstab
```

Add the following lines:

```
LABEL=ybdata  /mnt/ybdata auto defaults 00
```

```
/mnt/ybdata/ingres/ckp  /opt/Actian/VectorVW/ingres/ckp  none  bind,_netdev  0 0
```

```
/mnt/ybdata/ingres/data  /opt/Actian/VectorVW/ingres/data  none  bind,_  0
```

			netdev	0
/mnt/ybdata/ingres/dmp	/opt/Actian/VectorVW/ingres/dmp	none	bind,_netdev	0 0
/mnt/ybdata/ingres/jnl	/opt/Actian/VectorVW/ingres/jnl	none	bind,_netdev	0 0
/mnt/ybdata/ingres/log	/opt/Actian/VectorVW/ingres/log	none	bind,_netdev	0 0
/mnt/ybdata/ingres/work	/opt/Actian/VectorVW/ingres/ work	none	bind,_netdev	0 0
/mnt/ybdata/areplicate/data	/opt/attunity/replicate/data	none	bind,_netdev	0 0

5. Refresh the mounts to implement the changes on fstab:

```
mount -a
```

6. Change user ownership for actian and attunity users on the mounted file-system folders by running:

```
chown -R actian:actian /mnt/ybdata/ingres
```

```
chown -R attunity:attunity /mnt/ybdata/areplicate/
```

7. Remove the old db configuration:

```
rm /mnt/ybdata/ingres/data/vectorwise/vectorwise.db.conf
```

8. Create a new db configuration (where 4.x.x is the Hubble Accelerator version):

```
/opt/insightsoftware/hubble-accelerator/4.x.x/vector/
configurevector.sh
```

   For example:

```
/opt/insightsoftware/hubble-accelerator/4.2.1/vector/
configurevector.sh
```

9. Reboot.

Upgrading from a 4.x Version to another 4.x Version

1. Back up your data.
2. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

3. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

4. Stop the Hubble Appliance API Service:

```
systemctl stop hubble-applianceapi
```

5. Install the new version of Hubble Accelerator (follow the installation instructions in Install Hubble Accelerator).

Main Components of Hubble Accelerator

This section describes the procedures used when working with the main components of accelerator. Continue at:

- Actian Vector
- Qlik Replicate
- Hubble Backup API

Actian Vector

This section is divided into the following key areas for Actian Vector:

- Installation Path
- Basic Commands
  - Startup Vector
  - Shutdown Vector
- Maintenance Guide

Installation Path

Actian Vector will be installed by default on:

```
/opt/Actian
```

Basic Commands

- Start up Vector:

  ```
  systemctl start actian-vectorVW
  ```

- Shut down Vector:

  ```
  systemctl stop actian-vectorVW
  ```

- Optimizing the Accelerator Database
- sysmod - Modifying the System Catalogs
- Propagating Accelerator In-Memory Changes and Condensing the Log
- Disable new Connections to the Accelerator Database
- Rolling the Transaction Log
- Backing-up Tables
- Restoring Tables
- Adding New Database Users

Optimizing the Accelerator Database

The Accelerator database needs to be optimized to ensure the fastest execution paths are used for queries, which also has an impact on the amount of memory used for queries. The database vendor recommends that you optimize a table when the data has changed by 10%. After the initial optimization of the database, and taking the database vendors recommendation into consideration, we would recommend optimization on the database is performed weekly, for each schema. For high volume databases, the optimization will need to be performed more frequently.

By default, there are two schemas in the database, one for the replication data and one for Hubble. If you have created or are going to create new users/schemas for additional data, these will need to be added to the optimization tasks. Optimizing can be run at table level or schema level. See below for the instructions for both operations.

To use the optimizedb command as demonstrated in the following topics you must be logged on as the actian or root user.

Optimizing an Individual Table

Log onto the accelerator server and run the following command:

optimizedb -u<DB_UserName> <Database_Name> -r<TableName>

For example, if the default database is db, to optimize its f0911 table in the replication schema run the following:

optimizedb -ureplication -zfq db -rf0911

Optimizing all Tables in a Schema

Log onto the accelerator server and run the following command:

optimizedb -u<DB_UserName> -zfq <Database_Name>

For example, if the default database is db, to optimize all the tables in the replication schema run the following:

optimizedb -ureplication -zfq db

Example Output from optimizedb

Below is an example of the output that you would expect to see from running optimizedb:

```
[root@yb-vmware-pre-config ~]# optimizedb -uhubble -zfq db
I_OP091C        optimizedb: table 'activity_workspace_accepted' in database 'db' contains no rows.
I_OP091C        optimizedb: table 'activity_workspace_shared' in database 'db' contains no rows.
[root@yb-vmware-pre-config ~]#
```

Typically executions will be silent, returning the user to the command prompt when optimization has completed. However, you will be notified of any tables that contain no rows.

Scheduling Optimization

Cron can be used to schedule the optimization.

sysmod - Modifying the System Catalogs

As well as optimizing the Accelerator database, it is recommended that the sysmod command is periodically run. The sysmod command sets the system catalogs of a database to their currently defined storage structure. Doing so removes the overflow and deleted pages, which results in accelerating query processing.

To use the sysmod command as demonstrated in the following sections you must be logged on as the actian or root user.

Running sysmod for the Hubble Database

The sysmod operation requires exclusive access to the database. To ensure you have exclusive access to the database, you will need to follow the instructions in the following order:

1. Shutdown Replication.
2. Disable new connections to the Accelerator Database.

   For details of these procedures, refer to Chapter 5: Accelerator Shutdown Instructions of the Hubble Accelerator Maintenance Guide.

   Log onto the accelerator server and run the following command:

   sysmod <database>

   For example, the default database is db, run:

   sysmod db

   Then manually restart the Replication Service. For details of this procedure, refer to Chapter 6: Accelerator Startup Instructions of the *Hubble Accelerator Maintenance Guide*.

   The replication tasks can be restarted in two modes, Resume and Reload:

- Resume will read the source database log and resume reading the changes from where it left off.
- Reload will reload the data from the source database and then read the changes from the source database log.

Propagating Accelerator In-Memory Changes and Condensing the Log

The highly performance columnar database that drives Hubble Accelerator achieves its high performance by doing as much work as possible in memory, notably transactions are only written to disk when the system is quiet. The Actian system will attempt to propagate the transactions and condense this log periodically itself but on a system that is continually processing transactions this can fail. As such one of the key maintenance tasks for accelerator administrators is to monitor the volume of in-memory changes to ensure that it doesn't grow too large and ensure that when the accelerator is stopped, e.g. for upgrade, that it is done in such a manner that all the database transactions are written to disk.

Failure to do so could result in delays in startup as the unpropagated in-memory changes are read into memory on startup or the system failing to start if the in-memory changes have grown beyond the size of the memory allowed for the Actian process.

This is a 2-stage process:

- Propagate in-memory changes writes the in-memory changes to the database.
- Condense the log to remove the in-memory changes now written back to the database. Additional information relating to accelerator database log propagation can be found in the appendix: Appendix 1: Additional Information Relating to Log Propagation.

Propagating the in-memory changes and condensing the log can take some time. This will fail if in-memory changes are made via queries while this is running. As such you must ensure that the steps in the previous sections to effectively stop all queries running on the database must be carried out to ensure that this will complete successfully. This activity, given the time that it is likely to take must be planned at a time when it is least disruptive to the users of the system.

How to Monitor the size of In-Memory Changes

The command vwinfo -M <DATABASE> will return what tables are accumulating in memory change and the size of these.

Thus on a typical Accelerator the steps will be:

1. Connect to Hubble accelerator server.
2. Switch user to the Accelerator database user.

   # su actian
3. Run the above vwinfo command.

   # vwinfo -M db

How to Propagate the In-Memory Changes into the Log and Condense the Log

Proceed as follows:

1. Connect to Hubble accelerator server.
2. Switch user to the Accelerator database user.

   # su actian
3. At this point you need to decide if you want to propagate specific tables where the in-memory changes are particularly large or if you want to do this for all tables in the schema. As noted above, propagating changes can take some time and as the administrator you need to know how large a downtime window your business can tolerate and plan accordingly.

To propagate the in-memory changes for specific tables you need to run the command echo "call vectorwise (combine '<TABLE>')\g" for each table.

echo "call vectorwise (combine 'jde.f0911')\g" | sql db

Alternatively, to propagate the in-memory changes for all tables in the database run the following command.

echo "call vectorwise (combine)\g" | sql db

4. After propagating the changes to all the desired tables in a given schema the following command should be run to condense the log.

   echo "call vectorwise (condense_log)\g" | sql db

Disable new Connections to the Accelerator Database

As per the instructions for Shutdown Replication (see Chapter 5: Accelerator Shutdown Instructions of the Hubble Accelerator Maintenance Guide) propagating the log can fail if transactions are added while propagation is in progress. Apart from replication, we need to also stop new SQL sessions being started on the accelerator database; c.f. Propagating Accelerator In-Memory Changes and Condensing the Log.

1. Connect to Hubble accelerator server.
2. Switch user to the Actian database user.

   # su actian
3. Use iinamu to get the server number that you will need to pass to iimonitor.
4. # iinamu

   Actian Vector NAME SERVICE MANAGEMENT UTILITY --

   -- Copyright (c) 2009 Actian Corporation IINAMU> show

   INGRES * 41271 (sole server) IINAMU> quit
5. Run iimonitor using the number of the "sole server" returned by the above iinamu command.

iimonitor 41271 IIMONITOR>

6. Stop new SQL sessions from starting.

   IIMONITOR> set server closed User connections now disabled IIMONITOR>

7. Wait a few minutes to allow user sessions to naturally terminate and then get a formatted list of all the active sessions. The following only shows the details of a single session but will give you a feel for what to expect to see.

   IIMONITOR> show sessions formatted

   Session 0000000007023C80:1399772928 (jde810a) cs_state: CS_EVENT_WAIT (BIOR) cs_ mask: CS_INTERRUPT_MASK,CS_NOXACT_

   MASK OS_tid: 31151

   DB Name: db (Owned by: root) User: jde810a (jde810a) Session started at 3-May-2016 12:04:56 as user jde810a

   Terminal: batch Group Id:

   Role Id:

   Application Code: 00000000 Current Facility: CLF (00000001) Client user: Edward

   Client host: edward0912 Client tty:

   Client pid:

   Client connection target: db

   Client information: user='Edward',host='edward0912',conn='db', server='57468',session='1c250c0'

   Description:

   Query:

   Last Query: open ~Q cursor for SELECT CVCRCD, CVCDEC, CVDL01 FROM jde810a.F0013 for readonly

   Session 000000000714BFC0:-566962432 (integration_test) cs_state: CS_EVENT_WAIT (BIOR) cs_mask: CS_INTERRUPT_MASK,CS_NOXACT_

   MASK OS_tid: 4031

   ...

8. Shutdown each non-terminated session using the following command.

The session id that needs to be passed is the hexadecimal number that precedes the colon. Thus for the above example, Session 0000000007023C80:1399772928, the session identifier is 0000000007023C80.

IIMONITOR> remove 0000000007023C80

Session 0000000007023C80 removed

8. Repeat steps 6 and 7 until there are no non-administrator sessions running.
9.
10. Re-enable connections to the database. This is to ensure that you can run the sql command in the next section to allow the in-memory changes to the database to be propagated and condensed.

    IIMONITOR> set server open User connections now allowed

11. Exit iimonitor leaving the server closed.

    IIMONITOR> quit #

The iimonitor command is an extremely useful tool for monitoring and understanding what the accelerator database is doing. For more information on this tool refer to Appendix A of the Actian Vector 5.0 User Guide.

Rolling the Transaction Log

The vectorwise.log file, stored in /opt/Actian/VectorVW/ingre/files, accumulates all log messages for the accelerator database and this can grow quite large. This should be regularly rolled over to keep the size of the log manageable.

How to Check the size of the vectorwise.log File

Run the following command as the Actian user:

ls -lh /opt/Actian/VectorVW/ingres/files/vectorwise.log

How to Roll the vectorwise.log File

If the file is getting large it is advisable to roll the log file. The following command should be run as the Actian user.

echo "call vectorwise(vwlog_rotate)\g" | sql db

The expected output would be something along the lines of:

```
TERMINAL MONITOR Copyright 2015 Actian Corporation
Vector Linux Version VW 4.2.2 (a64.lnx/228) login
Fri Nov  3 03:24:47 2017
Enter \g to execute commands, "help help\g" for general help,
"help tm\g" for terminal monitor help, \q to quit

continue
* Executing . . .

continue
*
Your SQL statement(s) have been committed.

Vector Version VW 4.2.2 (a64.lnx/228) logout
Fri Nov  3 03:24:47 2017
```

Check the log size again and it should be much less in size. There should also be another file next to it named vectorwise.log-<today's date>. This can be safely deleted.

Backing-up Tables

Log onto the accelerator server and check that you have enough space to take backups of data.

The backup process creates uncompressed copies of the tables, and hence will consume disk space. Best practice recommendation is to create an NFS mount or similar to a remote server to store the backup of your accelerator data.

1. Create a directory to hold the backups:

   mkdir /tmp/<Backup_Directory_Name>

2. Navigate to the new directory:

   cd /tmp/<Backup_Directory_Name>

3. Optionally, if required you can create subdirectories for each schema you are backing up.

   The command that is used to take a backup of the table and data is called copydb. The function will create two files, copy.in and copy.out. These files are then used to create the export data files (copy.out) and then to restore the tables and data (copy.in).

   The copydb command takes the following parameters:

   - To copy all the tables in the schema owned by the database user:

     copydb <database_name> -u<DB_UserName>

   - To copy one or multiple tables from the schema owned by the database user:

     copydb <database_name> -u<DB_UserName> -r<TableName1> <TableName2>

4. As previously explained, running the copydb command will create two files, copy.in and copy.out. If I wanted to backup TableA from the replication schema, I would run:

   copydb db -ureplication -rTableA

This will create copy.in and copy.out into the directory where I ran the command.

5. I would then run the export command to generate the data file backup, based on the copy.out file. The syntax for the export command is:

   sql <database_name> -u<DB_UserName> < copy.out

   For example:

   sql db -ureplication < copy.out

   This will create individual data files named <tablename>.<schema> in the directory that where I ran the command from. In our example, TableA.replication.

Restoring Tables

1. Log on to Accelerator.
2. Copy the backup file onto the server (SCP) and into the directory that the backup was created from (see Hubble Accelerator Database Table Backup section).
3. Decompress the file if required.

   If the location of the files is different to the location the backup was made from you will need to edit the copy.in file and change the location specified in the file to the new location of the backup files. The copy.in file contains sections for Creating Tables, Grant Select and Copying Data, if you want to restore the data, but the table already exists in the database, then you will need to edit the copy.in file accordingly.
4. Then to use the copy.in file to import the data run:

   sql <database_name> -u<DB_UserName> < copy.in

For example:

sql db -uhubble < copy.in

Adding New Database Users

The ybdbsetup helper scripts is used to create a new db user.

A menu explaining how to use the script can be produced by running the following, without parameters:

[root@******]# ybdbsetup

Possible Argument for /usr/bin/ybdbsetup To create the database use db

e.g. ybdbsetup dbFor application user app_user -u -p/--password

e.g. ybdbsetup app_user -u=hubble -p=PassworD

For replication user use rep_user -u -p/--password

e.g. ybdbsetup rep_user -u=replication -p=PassworD For attunity user use att_user

e.g. ybdbsetup att_user

You only need to use this script to create extra users.

The difference between the app_user and rep_user parameters is that the app_user will be setup with idle disconnect timeout setup.

For example, to create a new app user called hubble2 with a password of mypassword enter the following:

ybdbsetup app_user -u=hubble2 -p=mypassword

Optimizing the Accelerator Database

The Accelerator database needs to be optimized to ensure the fastest execution paths are used for queries, which also has an impact on the amount of memory used for queries. The database vendor recommends that you optimize a table when the data has changed by 10%. After the initial optimization

of the database, and taking the database vendors recommendation into consideration, we would recommend optimization on the database is performed weekly, for each schema. For high volume databases, the optimization will need to be performed more frequently.

By default, there are two schemas in the database, one for the replication data and one for Hubble. If you have created or are going to create new users/schemas for additional data, these will need to be added to the optimization tasks. Optimizing can be run at table level or schema level. See below for the instructions for both operations.

To use the `optimizedb` command as demonstrated in the following topics you must be logged on as the `actian` or `root` user.

To optimize an individual table

Log on to the Accelerator server and run the following command:

```
optimizedb -u<DB_UserName> <Database_Name> -r<TableName>
```

For example, if the default database is `db`, to optimize the `f0911` table in the `replication` schema run the following:

```
optimizedb -ureplication -zfq db -rf0911
```

To optimize all the tables in a schema

Log on to the Accelerator server and run the following command:

```
optimizedb -u<DB_UserName> -zfq <Database_Name>
```

For example, if the default database is `db`, to optimize all the tables in the `replication` schema run the following:

```
optimizedb -ureplication -zfq db
```

Example Output from `optimizedb`

Following is an example of the output that you would expect to see from running `optimizedb`:

```
[root@yb-vmware-pre-config ~]# optimizedb -uhubble -zfq db
I_OP091C        optimizedb: table 'activity_workspace_accepted' in database 'db' contains no rows.
I_OP091C        optimizedb: table 'activity_workspace_shared' in database 'db' contains no rows.
[root@yb-vmware-pre-config ~]#
```

Typically executions will be silent, returning the user to the command prompt when optimization has completed. However, you will be notified of any tables that contain no rows.

Scheduling Optimization

Cron can be used to schedule the optimization.

`sysmod` - Modifying the System Catalogs

To optimize the Accelerator database, you should periodically run the sysmod command. This command modifies the system catalogs of your database to their current storage structure. By doing so, you remove overflow and deleted pages, which accelerates query processing.

To use the `sysmod` command as demonstrated in the following sections you must be logged on as the `actian` or `root` user.

To run `sysmod` over the Hubble database

Note: The `sysmod` operation requires exclusive access to the database

To ensure you have exclusive access to the database, you will need to follow the instructions in the following order:

1. Follow the instructions given in Shutdown Replication.
2. Follow the instructions given in Disable new connections to the Accelerator Database.

   Log on to the Accelerator server and run the following command:

   ```
   sysmod <database>
   ```

   For example, the default database is `db`:

   ```
   sysmod db
   ```

   Follow the instructions given in Manual Replication Service Startup

   The replication tasks can be restarted in the following two modes:

   - **Resume:** will read the source database log and resume reading the changes from where it left off.
   - **Reload:** will reload the data from the source database and then read the changes from the source database log.

Propagating Accelerator In-Memory Changes and Condensing the Log

This section covers instructions on how to manage in-memory changes and the transaction log for the Accelerator database. It includes steps for monitoring the size of in-memory changes, propagating these changes to the database, and condensing the log to ensure optimal performance and stability.

Introducing in-memory changes and the transaction log

To ensure high performance, the columnar database driving Accelerator performs most tasks in memory. Transactions are only written to disk when the system is idle. The Actian system tries to

propagate transactions and condense the log periodically, but this can fail if the system is constantly processing transactions.

As an Accelerator administrator, you need to monitor the volume of in-memory changes to prevent them from growing too large. When stopping the accelerator, for example, for an upgrade, make sure all database transactions are written to disk. Failure to do so can cause startup delays as unpropagated in-memory changes are read into memory, or the system may fail to start if the in-memory changes exceed the allowed memory size for the Actian process.

This is a two-stage process:

1. **Propagate in-memory changes:** Write the in-memory changes to the database.
2. **Condense the log:** Remove the in-memory changes now written back to the database. For more details, refer to Appendix 1: Additional Information Relating to Log Propagation.

> **Note:** Propagating the in-memory changes and condensing the log can take some time. This will fail if in-memory changes are made via queries while this is running. As such you must ensure that the steps in the previous sections to effectively stop all queries running on the database must be carried out to ensure that this will complete successfully. This activity, given the time that it is likely to take must be planned at a time when it is least disruptive to the users of the system.

How to Monitor the size of In-Memory Changes

The command **`vwinfo -M <DATABASE>`** will return what tables are accumulating in memory change and the size of these.

Thus on a typical Accelerator the steps will be:

1. Connect to Hubble Accelerator server.
2. Switch user to the Accelerator database user.

```
# su actian
```

3. Run the above vwinfo command.

```
# vwinfo -M db
```

How to Propagate the In-Memory Changes into the Log and Condense the Log

Proceed as follows:

1. Connect to Hubble Accelerator server.
2. Switch user to the Accelerator database user.

```
# su actian
```

3. At this point you need to decide if you want to propagate specific tables where the in-memory changes are particularly large or if you want to do this for all tables in the schema. As noted above, propagating changes can take some time and as the administrator you need to know how large a downtime window your business can tolerate and plan accordingly.

   To propagate the in-memory changes for specific tables you need to run the command **`echo "call vectorwise (combine '<TABLE>')\g"`** for each table.

   ```
   echo "call vectorwise (combine 'jde.f0911')\g" | sql db
   ```

   Alternatively, to propagate the in-memory changes for all tables in the database run the following command.

   ```
   echo "call vectorwise (combine)\g" | sql db
   ```

4. After propagating the changes to all the desired tables in a given schema the following command should be run to condense the log.

   ```
   echo "call vectorwise (condense_log)\g" | sql db
   ```

Disable new Connections to the Accelerator Database

> **Note:** According to the instructions for Shutdown Replication (see Hubble Accelerator Shutdown Steps), you must ensure that no transactions are added while propagating the log, as this can cause the process to fail. Besides stopping replication, you also need to prevent new SQL sessions from starting on the accelerator database during this time. Refer to the section on Propagating Accelerator In-Memory Changes and Condensing the Log for more details.

1. Connect to Hubble Accelerator server.
2. Switch user to the Actian database user.

   ```
   # su actian
   ```

3. Use `iinamu` to get the server number that you will need to pass to `iimonitor`.

   ```
   # iinamu
   ```

```
Actian Vector NAME SERVICE MANAGEMENT UTILITY --

-- Copyright (c) 2009 Actian Corporation
 IINAMU> showINGRES    *    41271 (sole server)
 IINAMU> quit
```

4. Run `iimonitor` using the number of the "sole server" returned by the above iinamu command.

```
# iimonitor 41271
IIMONITOR>
```

5. Stop new SQL sessions from starting.

```
IIMONITOR> set server closed
User connections now disabled
IIMONITOR>
```

6. Wait a few minutes to allow user sessions to naturally terminate and then get a formatted list of all the active sessions. The following only shows the details of a single session but will give you a feel for what to expect to see.

```
IIMONITOR> show sessions formatted
Session 0000000007023C80:1399772928 (jde810a    ) cs_state: CS_
EVENT_WAIT (BIOR) cs_mask: CS_INTERRUPT_MASK,CS_NOXACT_
MASK OS_tid: 31151
DB Name: db         (Owned by: root    ) User: jde810a
(jde810a    ) Session started at 3-May-2016 12:04:56 as user
jde810a
Terminal: batch Group Id:
Role Id:
Application Code: 00000000    Current Facility: CLF (00000001)
Client user: Edward
Client host: edward0912 Client tty:
Client pid:
```

```
Client connection target: db
Client
information:    user='Edward',host='edward0912',conn='db',
server='57468',session='1c250c0'
Description:
Query:
Last Query: open ~Q cursor for SELECT CVCRCD, CVCDEC, CVDL01
FROM jde810a.F0013 for readonly
Session 000000000714BFC0:-566962432 (integration_test    ) cs_
state: CS_EVENT_WAIT (BIOR) cs_mask: CS_INTERRUPT_MASK,CS_
NOXACT_
MASK OS_tid: 4031
...
```

7. Shut down each non-terminated session using the following command.

> **Note:** The sessionID that needs to be passed is the hexadecimal number that precedes the colon. Thus for the above example, Session 0000000007023C80:1399772928, the session identifier is 0000000007023C80.

```
IIMONITOR> remove 0000000007023C80
Session 0000000007023C80 removed
```

8. Repeat steps 6 and 7 until there are no non-administrator sessions running.

9. Re enable connections to the database. This is to ensure that you can run the sql command in the next section to allow the in-memory changes to the database to be propagated and condensed.

```
IIMONITOR> set server open
User connections now allowed
```

10. Exit `iimonitor`leaving the server closed.

```
IIMONITOR> quit #
```

The `iimonitor` command is an extremely useful tool for monitoring and understanding what the accelerator database is doing. For more information on this tool refer to Appendix A of the Actian User

Guide (Vector 6.3 guide).

Rolling the Transaction Log

The `vectorwise.log` file, stored in **/opt/Actian/VectorVW/ingre/files**, accumulates all log messages for the accelerator database and this can grow quite large. This should be regularly rolled over to keep the size of the log manageable.

How to check the size of the vectorwise.log File

Run the following command as the Actian user:

```
# ls -lh /opt/Actian/VectorVW/ingres/files/vectorwise.log
```

How to roll the vectorwise.log File

If the file is getting large it is advisable to roll the log file. The following command should be run as the Actian user.

```
# echo "call vectorwise(vwlog_rotate)\g" | sql db
```

The expected output would be something along the lines of:

```
TERMINAL MONITOR Copyright 2015 Actian Corporation
Vector Linux Version VW 4.2.2 (a64.lnx/228) login
Fri Nov  3 03:24:47 2017
Enter \g to execute commands, "help help\g" for general help,
"help tm\g" for terminal monitor help, \q to quit

continue
* Executing . . .

continue
*
Your SQL statement(s) have been committed.

Vector Version VW 4.2.2 (a64.lnx/228) logout
Fri Nov  3 03:24:47 2017
```

Check the log size again and it should be much less in size. There should also be another file next to it named `vectorwise.log-<today's date>`. This can be safely deleted.

Backing up Tables

Exporting tables from the database to the Linux server (Backup)

Log on to the accelerator server and check that you have enough space to take backups of data.

Note: The backup process creates uncompressed copies of the tables, and hence will consume disk space. Best practice recommendation is to create an NFS mount or similar to a remote server to store the backup of your accelerator data.

1. Create a directory to hold the backups:

   mkdir /tmp/<Backup_Directory_Name>

2. Navigate to the new directory:

   cd /tmp/<Backup_Directory_Name>

3. Optionally, if required you can create subdirectories for each schema you are backing up.

   The command that is used to take a backup of the table and data is called **copydb**. The function will create two files, copy.in and copy.out. These files are then used to create the export data files (copy.out) and then to restore the tables and data (copy.in).

   The copydb command takes the following parameters:

   - To copy all the tables in the schema owned by the database user:

     copydb <database_name> -u<DB_UserName>

   - To copy one or multiple tables from the schema owned by the database user:

     copydb <database_name> -u<DB_UserName> -r<TableName1> <TableName2>

4. Running the copydb command will create two files, copy.in and copy.out. For example, if you want to back up TableA from the replication schema, you would run:

   copydb db -ureplication -rTableA

   This will create copy.in and copy.out into the directory where I ran the command.

5. Run the export command to generate the data file backup, based on the copy.out file. The syntax for the export command is:

   sql <database_name> -u<DB_UserName> < copy.out

   For example:

   sql db -ureplication < copy.out

   This will create individual data files named <tablename>.<schema> in the directory that where I ran the command from. In our example, TableA.replication.

Restoring Tables

1. Log on to Accelerator.

2. Copy the backup file onto the server (SCP) and into the directory that the backup was created from (see Hubble Accelerator Database Table Backup section).

3. Decompress the file if required.

> **Note:** If the location of the files is different to the location the backup was made from you will need to edit the copy.in file and change the location specified in the file to the new location of the backup files. The copy.in file contains sections for Creating Tables, Grant Select and Copying Data, if you want to restore the data, but the table already exists in the database, then you will need to edit the copy.in file accordingly.

4. Then to use the copy.in file to import the data run:

   sql <database_name> -u<DB_UserName> < copy.in

   For example:

   sql db -uhubble < copy.in

Adding New Database Users

The ybdbsetup helper scripts is used to create a new db user.

A menu explaining how to use the script can be produced by running the following, without parameters:

[root@******]# ybdbsetup

Possible Argument for /usr/bin/ybdbsetup To create the database use db

e.g. ybdbsetup dbFor application user app_user -u -p/--password

e.g. ybdbsetup app_user -u=hubble -p=PassworD

For replication user use rep_user -u -p/--password

e.g. ybdbsetup rep_user -u=replication -p=PassworD For attunity user use att_user

e.g. ybdbsetup att_user

You only need to use this script to create extra users.

The difference between the app_user and rep_user parameters is that the app_user will be setup with idle disconnect timeout setup.

For example, to create a new app user called hubble2 with a password of mypassword enter the following:

ybdbsetup app_user -u=hubble2 -p=mypassword

Qlik Replicate

Contents:

- Qlik Replicate Installation Path
- Qlik Replicate Basic Commands
- Qlik Replicate Maintenance Guide

Qlik Replicate Installation Path

On the Hubble Accelerator server, Qlik Replicate is installed under the following directory:

/opt/attunity

The replication logs are created in the following directory:

/opt/attunity/replicate/data/logs

Qlik Replicate Basic Commands

- Startup Qlik Replicate:

  /etc/init.d/areplicate start

- Shutdown Qlik Replicate:

  /etc/init.d/areplicate stop

If there are tasks running on the system and you are unsure if they are still active follow the Qlik Shutdown Replication procedure.

- Accessing Qlik Replicate Console:

  Open a web browser and enter the following URL: https://[accelerator_IP]:3552/attunityreplicate

User: admin

Password: As detailed in Installing Hubble Accelerator the password is generated on install.

Qlik Replicate Maintenance Guide

Contents:

- Shutdown Qlik Replication
- Importing Replication Tasks
- Exporting Replication Tasks
- Using add-on to Remove Invalid Characters during Replication

Shutdown Qlik Replication

Propagating the transaction log to disk will fail if processes are making database changes. As such the replication service must be stopped before attempting to propagate the log.

Propagating Accelerator In-Memory Changes and Condensing the Log.

1. Connect to Hubble accelerator server.
2. Switch user to the attunity replication user.

   # su attunity

3. Get a list of all the running replication tasks.

   # ps -ef | grep repctl

```
[root@yb~]# ps -ef | grep repctl
attunity  2349     1  0 May24 ?        00:00:00 /opt/attunity/replicate/bin/repctl service start
attunity  2350  2349  0 May24 ?        00:43:51 /opt/attunity/replicate/bin/repctl service start
attunity  6660  2350  1 Nov20 ?        05:57:29 repctl reptasksrv PRODDTA 127.0.0.1:3552 2350
root     12585 12579  0 14:44 pts/0    00:00:00 grep repctl
attunity 22588  2350  0 Nov06 ?        00:18:24 repctl reptasksrv JDE812 127.0.0.1:3552 2350
attunity 23203  2350  0 Nov06 ?        01:44:48 repctl reptasksrv CUSTOM_TABLES 127.0.0.1:3552 2350
```

The running replication tasks are the processes that start repctl reptasksrv <TASK_NAME>. In the example above there are 3 running tasks:

- PRODDTA
- JDE812
- CUSTOM_TABLES

4. For each task returned by the above, run the following command to get the status of the task and if it is running to stop the replication task. (You can also stop the tasks in the Qlik Console, but you should check the status as instructed below)

   # ./repctl connect\; gettaskstatus <TASK_NAME>\; disconnect # ./repctl connect\; stoptask <TASK_NAME>\; disconnect

The stoptask command used above will signal the task to stop, if the task is currently showing latency, then the task will not stop immediately but will stop after processing the changes that are causing the latency. This will allow you to restart the task with the resume feature, this means you do not need to reload the data. If you run the stoptask command and manually kill the associated replication process, then you will not be able to use the resume command to restart the task, you will have to restart the task using the reload functionality in order to ensure data consistency. All the tasks must be stopped before moving on to Step 5 and 6.

5. Shutdown Qlik replication service. This will prevent users form connecting remotely using Qlik Console and starting new tasks.

   # ./S65areplicate stop

6. Confirm that the services has stopped by checking that there are no repctl processes running:

   # ps -ef | grep repctl

Import Replication Tasks

If you are importing a backup file, the backup file will also contain the connection details (encrypted). After the import process has been completed, the password used for each of the database connections is removed. The password will need to be re-entered in the manage database connection dialog for each connection.

1. Copy the backup file (json) onto the accelerator server in the Linux directory:

   /opt/attunity/replicate/data/imports

2. Log onto the accelerator server and change the owner of the backup file to the user “attunity”, if necessary:

   chown attunity:attunity <FileName.json>

3. Log in or change the accelerator user account that you are logged in as to the “attunity” user:

   su - attunity

4. To import the backup file (json) by entering the following:

   repctl importrepository json_file=<Name_of_Json_File>

For example, for a json file named Ora_Rep_F0911.json, run the following command:

repctl importrepository json_file=Ora_REP_F0911

The name of the file is case sensitive.

You do not need to enter the .json extension.

Export Replication Tasks

1. Log onto the Accelerator server and change directory to /opt/attunity/replicate/data/imports:

   cd /opt/attunity/replicate/data/imports

2. Change the accelerator user account to use the attunity user:

   su - attunity

3. To export an individual task, enter the following:

   repctl exportrepository task=<name_of_task>

For example, if the task is called PRODDTA, enter (note the task name is case sensitive):

repctl exportrepository task=PRODDTA

4. To export all tasks, enter the following:

   repctl exportrepository

A json file will be created in the /opt/attunity/replicate/data/imports directory. This can now be copied from the Accelerator (SCP) and to your backup location.

```
[root@yb-banfi-1 bin]# ./repctl exportrepository
command exportrepository response:
{
        "message":      "Export succeeded, export location: /opt/attunity/replicate/data/imports"
}
[exportrepository command] Succeeded
```

Using add-on to Remove Invalid Characters during Replication

Summary

This article describes a solution dealing with characters in the ERP source tables that are considered invalid during the replication to Vector and cause the replication task to fail with an error similar to the following:

VWLOAD output: loading Error in file '/opt/attunity/replicate/data/tasks/badora/vectorwise/1/LOAD00000001.csv', record 1: Illegal character encountered in input Input record: '"@@?@"' processed 1 records, loaded 0 records, 1 errors

The specific case addressed is when text columns representing descriptions in the ERP system were containing characters that are not UTF-8. A user-defined function called fix_latin was added to the transformation functions. This function accepts a string of characters with mixed encodings, for example it may contain predominantly UTF-8 characters but may also contain characters encoded with Windows-1252 or ISO 8859-1. It will create a UTF-8 output that is based on the following conversions:

0x00 - 0x7F ASCII - passed through unchanged

0x80 - 0x9F Converted to UTF8 using Windows-1252 (aka CP1252) mappings 0xA0 - 0xFF Converted to UTF8 using ISO/IEC 8859-1 (aka Latin-1) mappings

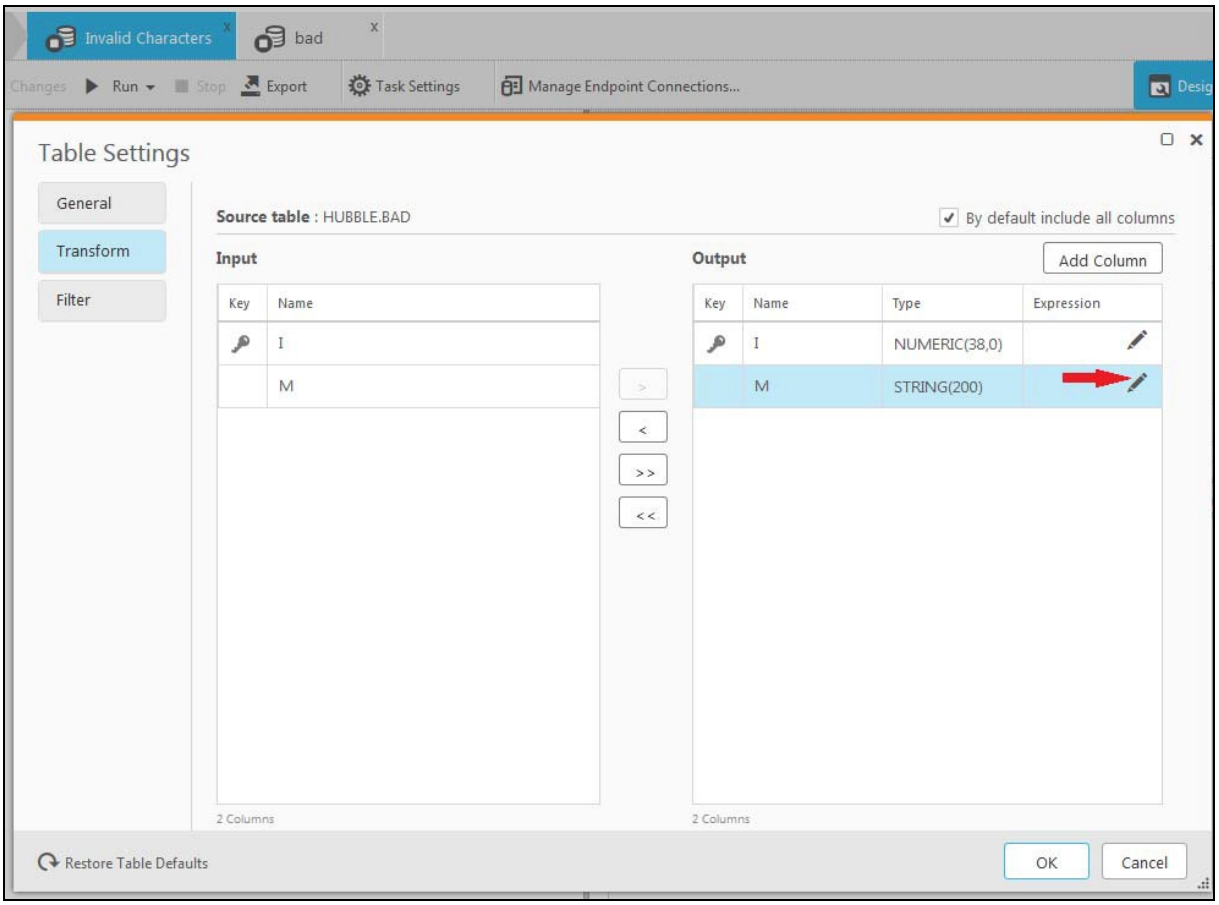
Five characters that do not have a mapping are converted to a string %xx with xx being the hexadecimal value, for example 0x81 is mapped to %81. Multi-byte UTF8 characters will be passed through unchanged, although over-long UTF8 byte sequences will be converted to the shortest normal form (see the original documents for the perl module).

Note that this solution has been designed and tested using Oracle as the source database. The fix_latin function is also available when other source database types are being used but it is untested and unsupported in that scenario.

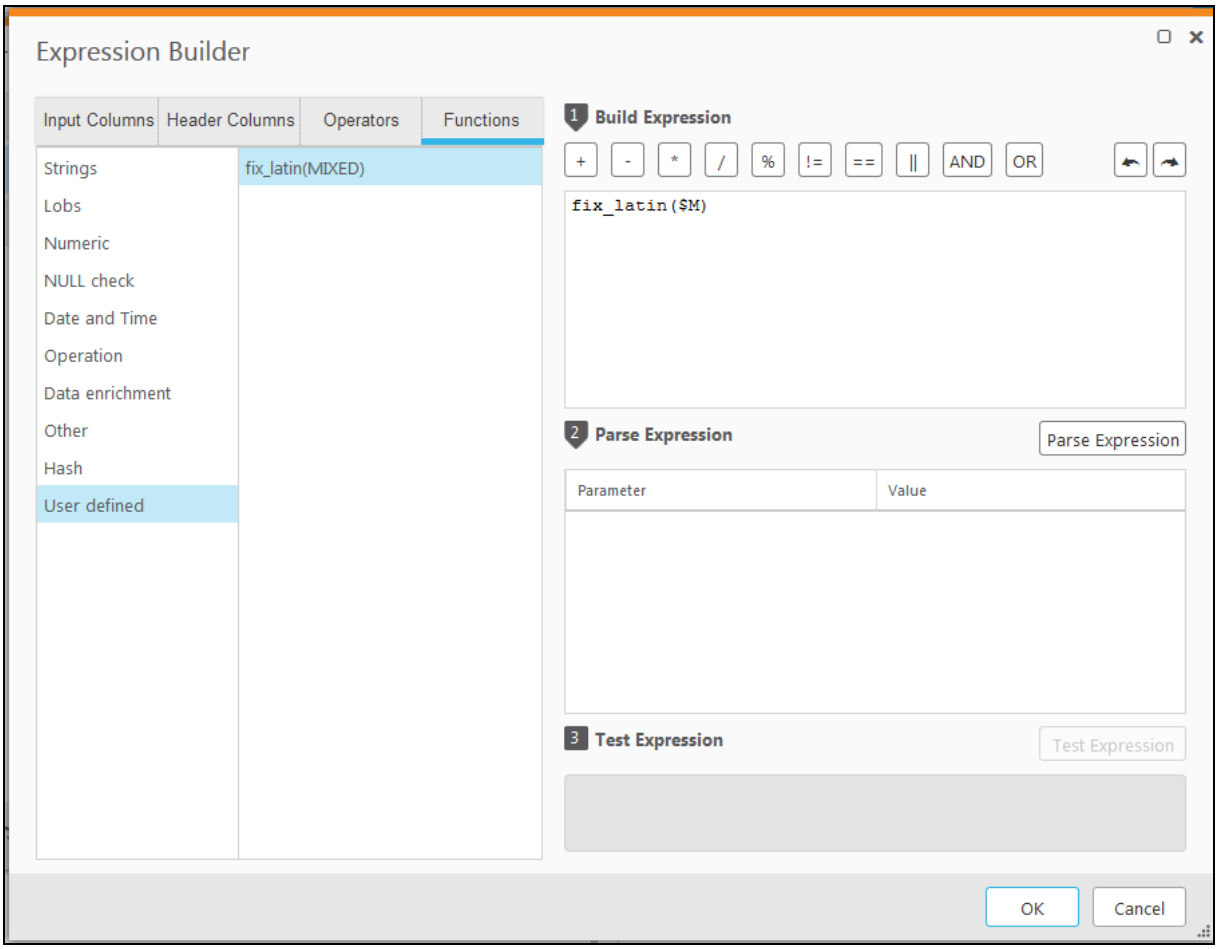
Manual Configuration Steps for Qlik Replicate

For any column that has invalid characters, a custom transformation can be set up in Qlik Replicate to replace these characters.

1. In the Table Settings, on the Transform tab, select the pen symbol to edit the transformation expression for the column in question:



2. Use the fix_latin function available on the Functions tab under User defined. Reference the source column that needs to be transformed as input to the function (M in the example below).



3. Repeat above steps for all text columns that have the same issue.

Hubble Backup API

Contents:

- Hubble Backup API Installation Path
- Hubble Backup API Basic Commands
- Hubble Backup API Vector Maintenance Guide

Hubble Backup API Installation Path

Hubble Backup API will be installed by default on:

/sr/libexec/hubble-applianceapi

Hubble Backup API Basic Commands

- Start up:

  systemctl start hubble-applianceapi

- Shut down:

  systemctl stop hubble-applianceapi

Hubble Backup API Vector Maintenance Guide

Contents:

Backing up the Accelerator

The accelerator backup feature provides the ability to back up the data on Hubble Accelerator which has not been replicated using Qlik Replicate. This mainly comprises data submitted as part of planning and budgeting.

The data replicated from your ERP is not backed up as the replication process will be used to re-import that data. The backup includes the Qlik repository and all database objects and data stored under the "hubble" Vector database schema.

Storage of Backup Archives

Backup archives are written to the location /import/dbbackup on the Accelerator.

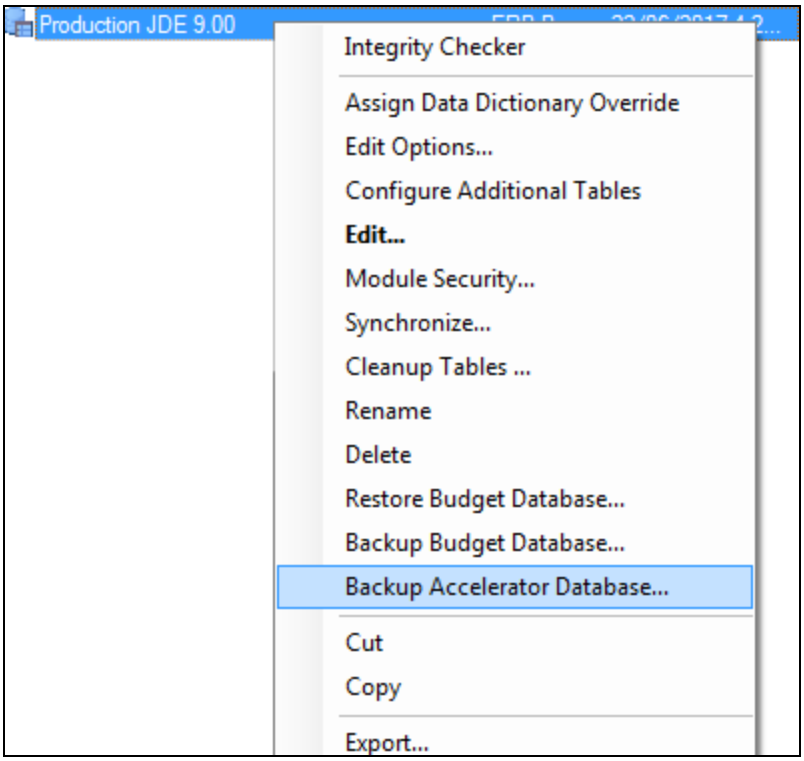
Do not rely on storing the backup files on the same physical server as the accelerator. It is recommended that the backup folder be mounted on an external device, e.g. using NFS. This NFS mount must be configured before a backup is run.

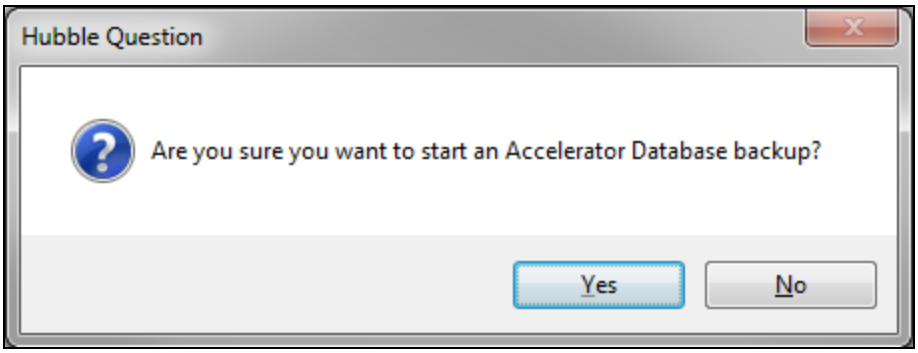
If the folder /import/dbbackup does not exist then any backup operations will fail.

Manually Triggering a Backup

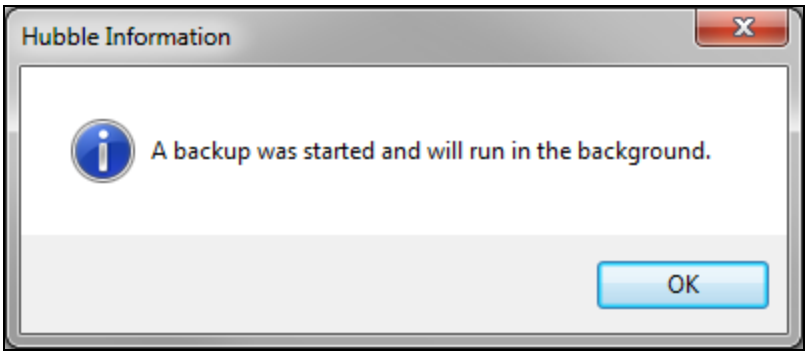
Note that the backup runs for the schema called “hubble” on the Accelerator ignoring the schema used on the profile.

The backup process is triggered from the Administration tool by right-clicking on a profile configured to use an Accelerator for its data connection.

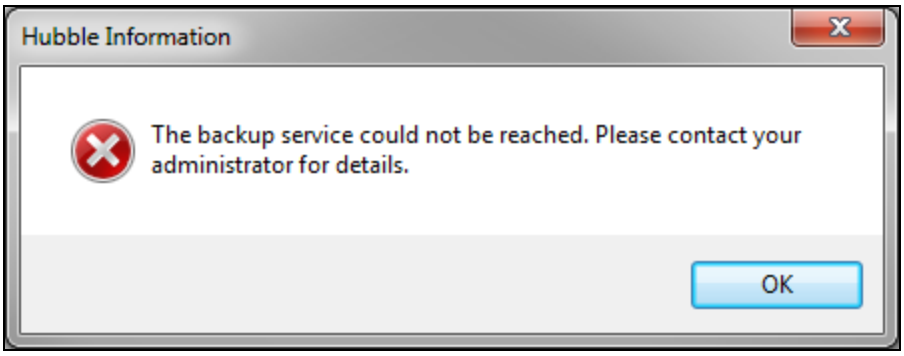




On clicking Yes the following message is presented:



If there was an error starting the backup (e.g. the Accelerator could not be reached) then this will be shown to the user as an error message:

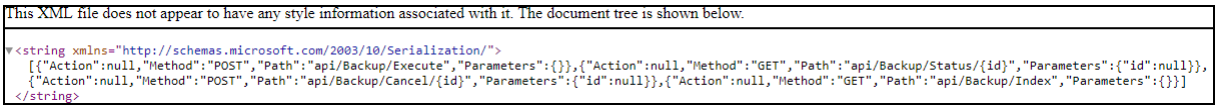


Any errors which occur during the backup process (e.g. insufficient space to store the resulting file) will not be reported back to the user. Instead these will be logged to the file / tmp/ybapi.log on the Accelerator.

Using the REST API to Trigger a Backup

The backup process has a public REST API which can be called using a scheduling tool which allows a URL to be called using the POST Method. The API URL is called using the accelerator server hostname. e.g. http:// hostname:5000/api/.

The API service can be tested by entering the API URL into a browser, the following should be returned:



REST API Details

The three backup operations that can be performed from the API are, starting a job, checking on its status, and canceling it.

Execute:

- Verb: POST
- URL: http://hostname:5000/api/Backup/Execute
- Parameters: None
-
- Result: Status JSON (see below)

Starts a new backup job and assigns it an ID that can be used for the following calls. The backup process will continue running after the HTTP call has completed.

Status:

- Verb: GET
- URL: http://hostname:5000/api/Backup/Status

- Parameters: 'id' - The ID from a previous status JSON result.
- Result: Status JSON (see below)

Returns the status of the backup job, based on the ID from a previous Execute operation.

Cancel:

- Verb: POST
- URL: http://hostname:5000/api/Backup/Cancel
- Parameters: 'id' - The ID from a previous status JSON result.
- Result: Status JSON (see below)

Attempts to cancel the backup job identified by the ID from a previous Execute operation.

Status JSON:

{

"Id": "b61a96c2134f4a59b91188c94e24aa57",

"Status": "Queued",

"Message": "Job was queued for execution."

}

The result of all three calls is a JSON object in the same format and with the same three elements:

- Id: The ID of the job. For Execute this will be the ID that must be saved for future information. For

Status or Cancel calls this will echo back the ID supplied to the request.

- Status: There are six possible values for this field:
  - Success - The backup job was completed successfully
  - Queued - The backup job has been queued and is waiting to be processed
  - Executing - The backup job is being executed.
  - Canceled - The backup job has been canceled. This is unlikely to be received in practice as canceled jobs are removed from the queue once they have completed.
  - LimitReached - An internal limit (e.g. number of concurrent jobs) has been reached that meant that the job could not be started.
  - Error - An internal error has occurred. As well as the message, further information will be available in the accelerator API's internal log file.

- Message: A human-readable explanation of the status. This may provide more information, especially in the case of an error.
-

Scheduling calls to the Accelerator Backup REST API

You may also already possess scheduling software which is capable of calling REST APIs directly. If not, below are two example approaches to scheduling the backup using the API. There is widespread guidance online regarding the use of these tools - Hubble cannot provide specific support for them.

Windows Task Scheduler and PowerShell

You can schedule a PowerShell script to invoke the backup API using Windows Task Scheduler. The script should look like the following:

Invoke-RestMethod -Uri "http://<hostname>:5000/api/Backup/Execute" -Method Post

where you replace <hostname> with the host name or IP address of your accelerator. To learn how to schedule this see for example https://social.technet.microsoft.com/wiki/contents/articles/38580.configure- to-run-a-powershell-script-into-task-scheduler.aspx.

To run unsigned PowerShell scripts, you may need to relax the 'execution policy' on the server, or override it on the PowerShell command line. See for example https://superuser.com/questions/106360/how-to-enable- execution-of-powershell-scripts.

Linux Cron and Curl

You can schedule the Linux command 'curl' using the scheduler 'cron'. The command you need to schedule should look like the following:

curl -X POST -d "" http://<hostname>:5000/api/Backup/Execute

where you replace <hostname> with the host name or IP address of your accelerator. Depending on your Linux distribution, you might have to use 'wget' instead of 'curl'.

Cron is not available within an accelerator deployed as a VMWare or AWS image, but you can schedule the task on any other Linux server. If you have deployed the accelerator as a Docker container, then you can use cron in the Docker host server which you supplied.

Troubleshooting

1. Check the log file “/tmp/ybapi.log” on the accelerator for error messages:
    a. Job with a given ID was not found. – Either the Cancel or Status API commands were executed with a Job ID that is either invalid or has completed.
    b. Unable to move file: {Error} – Several steps involve moving files, one of these failed. Examples are moving the replication backup file to a temporary folder, or moving the backup file to '/import/ dbbackup'.
    c. An error occurred while executing system command: {Command}. Error: {Error} – While we ask Qlik or Vector to perform a backup their data an error could occur.

d. Not enough space is left in '{Location}': minimum required {xx}MB, space left {xx}MB. – Both the accelerator's temporary directory and '/import/dbbackup' must have a minimum amount of space available. By default this is 2GB.
e. Unable to determine the drive of path '{Location}' – While checking for available space there was an error determining the drive to check.
f. Directory '{Location}' does not exist. – The directory does not exist.
g. File {Location} does not exist. – The file does not exist. This may occur because of an internal error that prevents a file we're trying to move from being created.

2. Restart the service.

Restoring an Accelerator Backup

Introduction

The Accelerator backup will come in the form of a Gzipped Tar archive which contains the contents of the budgeting and activity data stored on the database's Hubble schema. It also contains a backup of the Qlik replication tasks.

The process of restoring a backup consists of copying the archive onto the Accelerator then restoring each portion in turn. The order in which these tasks are run is important as attempting to restore the Hubble schema before allowing Qlik to re-replicate is likely to fail.

It assumes an empty Accelerator as the destination and that you have access to the 'root' account of it.

Preliminary Steps

To restore these backups onto a fresh server it is necessary to connect to a terminal (SSH) session on the Accelerator as root, and to copy the backup archive onto it.

We will assume that a Windows machine with Putty installed is being used. A Linux desktop could also be used with minimal modification to the steps.

Copying the Backup

1. From the host machine this can be achieved with Putty PSCP command. Replace the location of backup file and the Accelerator's IP address or host name.

   pscp -scp path\to\hubble_schema_backup.tar.gz root@accelerator_ip:/tmp/

2. Once there the archive should be expanded using the following commands from within the root SSH session.

   cd /tmp

   gunzip hubble_schema_backup_2017_07_15_21_56_10.tar.gz tar xvf hubble_schema_backup_2017_07_15_21_56_10.tar

3. This will create two folders, AttunityBackup and DatabaseBackup.

The first of these contains the definition of the ERP database replication tasks, whilst the second contains the data from the ‘Hubble’ schema of the database. We need to give the relevant accounts full access to 'their' data,

chown -R attunity:attunity /tmp/AttunityBackup chown -R actian:actian /tmp/DatabaseBackup

Qlik Backup

Importing the Replication Definition

The 'AttunityBackup' directory should contain a single file called Replication_Definition.json which contains the configuration of the Qlik components.

The major parts of this file are the definitions of the tasks, and the database connections that feed them. This can be imported into the Accelerator’s Qlik database with the following commands in the SSH session.

su attunity

repctl importrepository json_file=/tmp/AttunityBackup/ Replication_Definition.json

Create Schemas on the Accelerator

Before we can execute the replication tasks it is necessary to create the schema(s) into which this data will be placed. This is performed from within a SSH session and must be performed for each destination schema required by an Qlik Replicate task.

1. From the SSH session execute the command below as a root user:

   sql iidbdb

2. From within the SQL terminal create a new user for the target schema and then quit the application:

   create user schemaName with password='password' \g grant all on database db to schemaName \g

commit \g

\q

3. Now we need to create and drop a table for the above user, to initialize the new schema. From the SSH session execute the following command:

   sql db -uschemaName

From within the SQL terminal:

create table t1 (c1 char(1)); \g commit \g

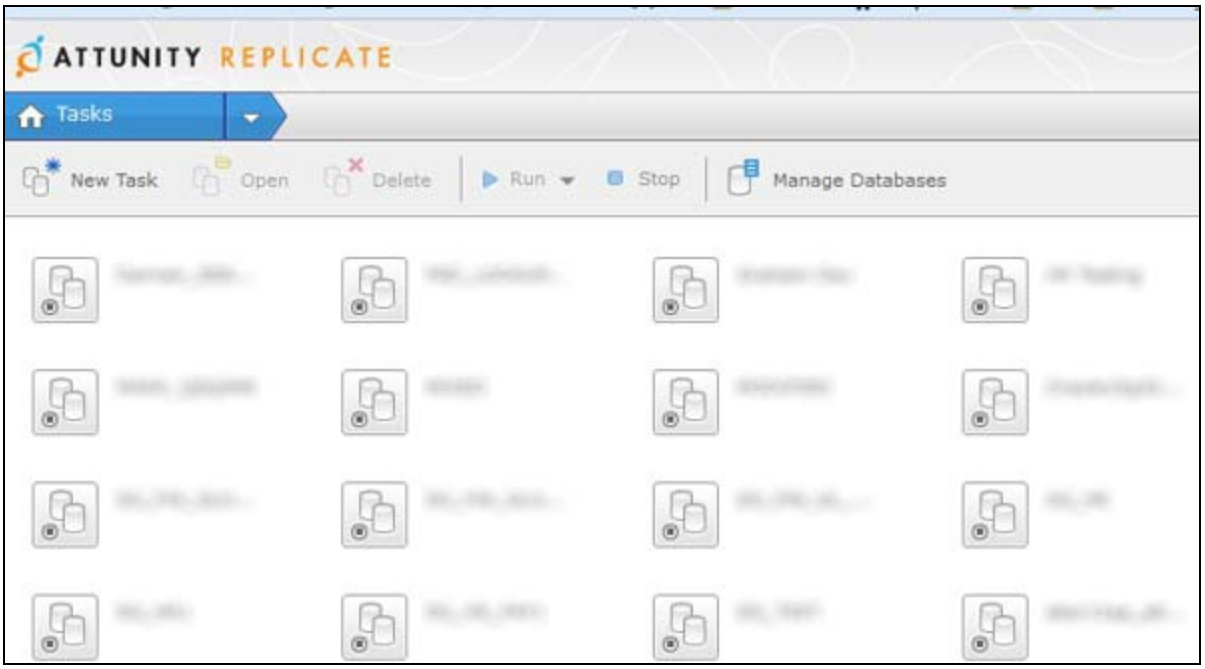
drop table t1; \g commit \g

\q

Run Qlik Replicate Tasks

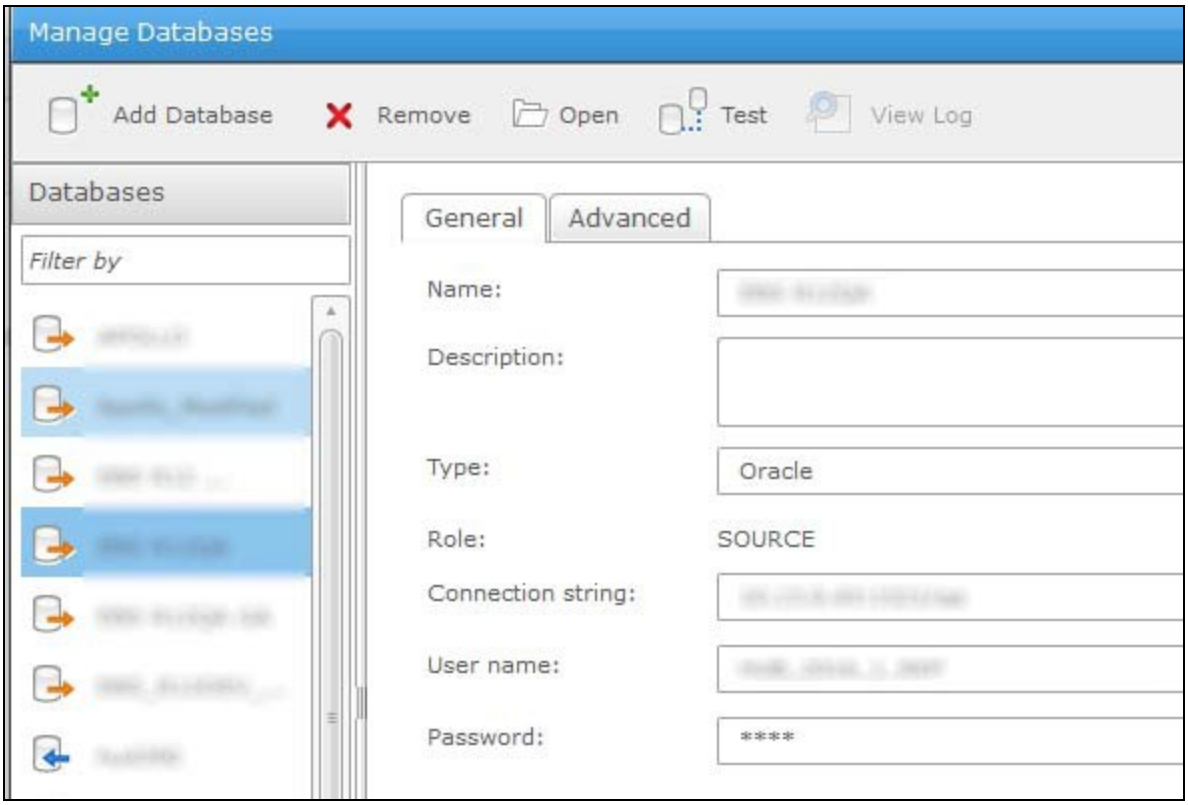
1. Configure the Qlik Replicate console to connect to the new Accelerator. This is described elsewhere.

2. Run Qlik Replicate Console and note that the tasks will be present.

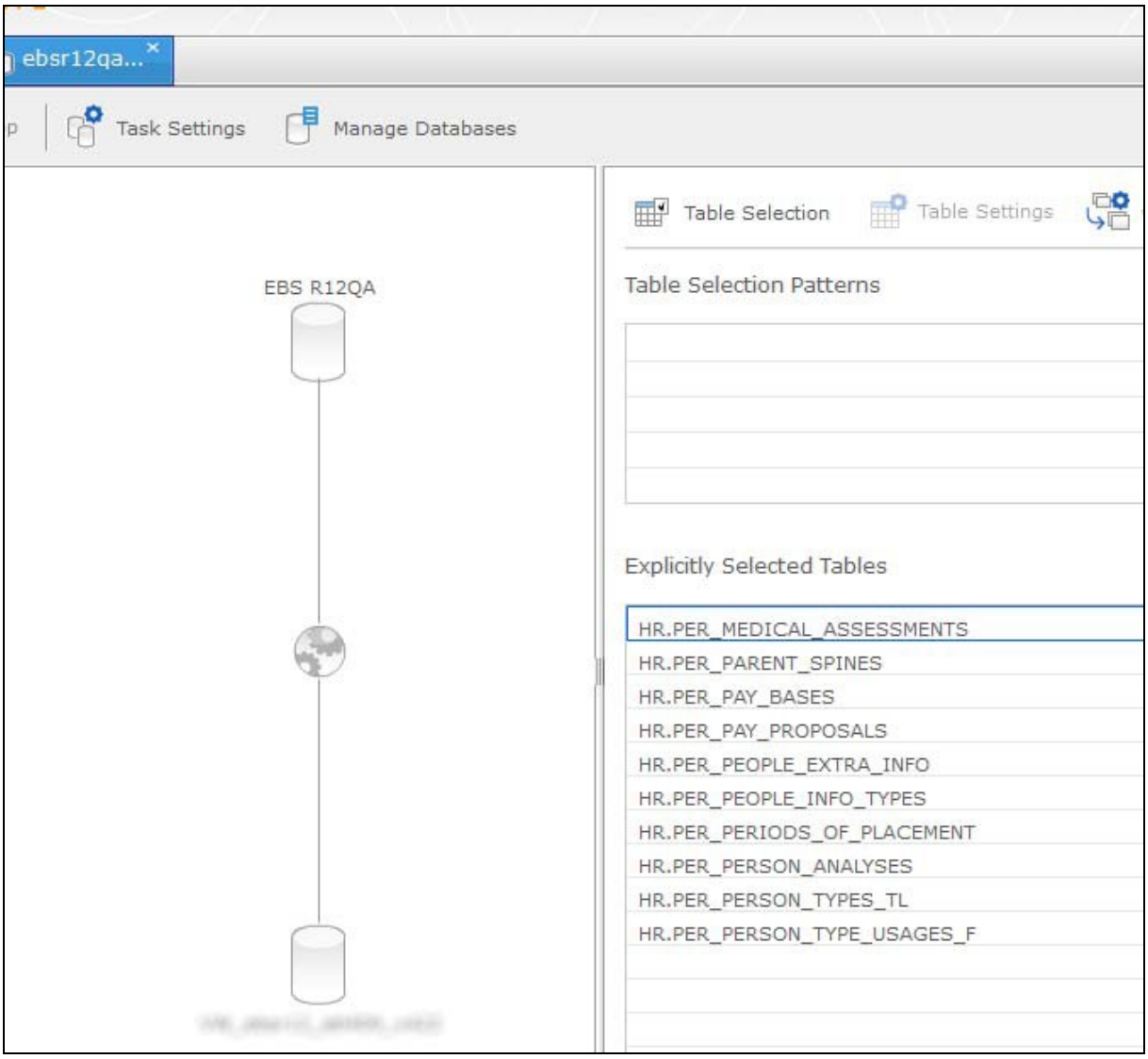


3. Click Manage Databases.

4. Click Open on the source/target databases and re-enter the passwords and test the connections.



5. Open the task we are interested in and check that there are tables populated in the Explicitly Selected Tables list.



If there are none, then it is likely that permissions on Linux were not given to the Attunity user earlier on (before repctl importrepository was run).

6. Run the task. Database Hubble Schema File Contents

   The DatabaseBackup folder contains a file for each of the tables in the Hubble database schema. This

   includes:

   - Budgeting data. Budgeting tables will be backed up including the custom strategic planning tables.

     Much of these tables are named with a two-letter prefix then ‘budget’. Also included in this group are ‘is_generic’ and ‘isbpayroll’ ('is' is the insightsoftware prefix)

- Activity tables. These tables' names all begin with 'activity' and contain user activity logged by the application.

- Dual table (for EBS only). This is a dummy table which may be present, but contains no data.

  In addition to these tables there will be two other files, called 'copy.in' and 'copy.out'. These are scripts to automate the insertion and extraction of data.

  Copy.in Script

  The 'copy.in' script is used to restore the exported data. As well as containing the definition of the tables exported it also contains links to the files containing the tables' contents.

  There are known scenarios where the backed up data contains synonyms that point to tables that are not included in the replication tasks. If this is the case then it is preferable to allow the restoration process to run through without stopping on error. The list of errors can then be examined after the import to verify the missing synonyms, or other errors that occurred.

  By default the import process will stop when it encounters an error. To allow this to continue it is necessary to edit the 'copy.in' file using a text editor such as nano. Near the top of the file will be the line

- \nocontinue

- To allow the import to continue on error this should be changed thus, and the file saved.

- \continue

- Performing Import

- It is required that the Hubble schema on a fresh Accelerator is empty. A freshly deployed Accelerator will have the five Activity tables created automatically, which must be deleted before restoring the backup. If the Activity tables are not deleted the restore script will fail when it attempts to recreate them.

- Start the restore process from the backup directory with:

- cd /tmp/DatabaseBackup

- sql db -uhubble < copy.in > output.txt

- This line assumes that the database is called 'db' and the user/schema is 'hubble'.

- Changes to the user used will require manual changes to the copy.in script as the original schema name is mentioned there.

- Checking for Errors

- The output of the restore process has been logged to 'output.txt'. It is therefore necessary to identify if there have been any errors logged to this file, using the following command:
- grep ^E_ output.txt
- If the import encountered tables are no longer replicated or had been created manually, but which had synonyms, then errors like this will be listed:
- E_US0845 Table 'ar_receipt_classes' does not exist or is not owned by you.
- The surrounding text for this error will detail the operation that was being attempted when the error occurred. This can be found by examining the log file in a text editor or viewer.

Hubble Accelerator Shutdown Steps

This section contains the instructions for safely shutting down Hubble Accelerator. The sequence should be:

1. Shutdown Qlik Replication.
2. Disable new Connections to the Accelerator Database that could write transactions.
3. Propagating Accelerator In-Memory Changes and Condensing the Log.
4. Rolling the Transaction Log.
5. Shutdown Actian Vector (see Actian Vector Basic Commands). The server can now safely be rebooted.