Managed File Transfer

In many organizations a large part of the exchange and distribution of information is realized by the copy and transfer of data files. As the number of files is ever increasing, the results are serious management and security issues. These issues are predominantly operational, so awareness at management level is typically low.

Consider the following use cases.

For years, corporations have been moving from bespoke to standard software: traditional locally installed packages or cloud delivered SAAS packages. Generally, standard software does not allow access to data directly but instead provides an application layer interface. For bulk transfers, the service / message style interfaces perform poorly, so file-based interfaces are common for these kinds of transfers. Files often contain the right information in the wrong format. Rather than script file changes and build an unmanageable library of scripts there is a need for a Managed File Transfer (MFT) product.

Recent trends lead to more data being stored in files:

- Hadoop's HDFS provides scalable, sheer unlimited capacity to store any files. Partly due to the limitations to update files history is often stored in completely separate files.
- More and more applications generate or use rich content like media (photos, videos) which is also commonly stored.

Corporations often use workplace software like SharePoint to formalize desktop processes and information. To link these processes to transactional systems there is a requirement to integrate these systems with workplace & office files, and with that the need for MFT. A SharePoint frontend on a transactional system also eliminates all kinds of unwanted direct access to business applications.

Challenges

FTP is a commonly used file transfer protocol but it offers no guarantees in delivery of files. To ensure delivery additional scripting is needed. As a result, every file transfer needs its own script, the number of scripts explodes and the environment becomes unmanageable. Creating a script is easy, maintaining and operating it is not.

To address those issues, organizations sometimes deploy a "file exchange platform" to manage these scripts. Due to the nature of scripting, this platform (often not more than a server with storage and lots of scripting) soon becomes a management nightmare itself. Most of the scripting is not documented, it is often unknown what exactly is happening and no real management is possible. Whenever there is an interruption, fixing scripts involves reverse engineering, long downtimes and even "hacking".

When file transfer to customers or partners and with that external transfers become necessary, the file exchange platform is often duplicated to the DMZ. Now there is not only a management problem, but also a security risk.

The end result is a deadly combination of high costs and high risks.

HVR for MFT

HVR provides Managed File Transfer functionality. This can be used as a tool in its own right, but also as part of the integrated HVR suite for enterprise data integration. Complex file transfer chains can be configured, scheduled and controlled from a central point on an enterprise-wide level. HVR supports 3 types of file transfer: file-to-file, database-to-file and file-to-database.

File-to-file transfer

An HVR file-to-file transfer will copy the files from one file location (the source location) to one or more other file locations (the target locations). A file location is a directory or a tree of directories, which can either be accessed through the local file system (Unix, Linux or Windows) or through a network file protocol (FTP, FTPs, sFTP or WebDAV). Files can be copied or moved. In the latter case, the files on the source location are deleted after they have been copied to the target locations. The file contents are normally preserved, but it is possible to include file transformations in the copy process using external commands or definitions defined in an XSLT file.

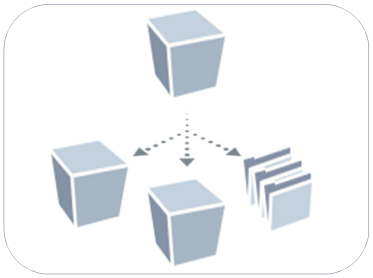
To distribute sets of files HVR provides the possibility to copy files selectively from the source location by matching their names to a predefined pattern. This feature also enables the routing of files within the same source location to different target locations on the basis of their file names to enable selective file distribution scenarios.

File-to-database transfer

In a file-to-database transfer data will be read from files in the source file location and replicated into one or more target databases. The source files are by default expected to be in a specific HVR XML format, which contains the table information required to determine to which tables and rows the changes should be written in the target database. It is also possible to use other input file formats by including an additional transformation step in the file capture. Support for CSV is available out-of-the-box, but any format can be handled by providing an external command or an XSLT definition.

Database-to-file transfer

In a database-to-file transfer the data is read from a source database and copied into one or more files on the source file location. By default the resulting files are in the HVR XML format preserving the table information. However, CSV is also supported out-of-the-box and other file formats can be created by including an additional transformation command or XSLT definition in the file output. As in the Continuous Database Replication between databases, it is possible to select specific tables and rows form the source database and convert names and column values.



Advanced scenarios

HVR's flexible architecture and seamless file integration make a combination of various scenarios also possible. For example combining distribution and conversion into a database distribution & file conversion channel. HVR's flexible architecture enables interchanging all kinds of scenarios easily!

HVR Managed File Transfer shares underlying data transport architecture with database refresh and replication, taking advantage of common features such as continuous data streaming, data compression and network encryption. As a result HVR MFT is optimized for maximum performance, efficiency and scalability making very effective use of network resources. It can easily handle multi-gigabyte files sent in a single task to or from 100 or more locations.

Supported sources	**Supported targets**
Oracle, all editions, including Amazon RDS	All supported sources
SQL Server, all editions, including Azure	Teradata
DB2 on Linux, Unix and Windows	Actian Vector (Vectorwise)
Ingres	Actian Matrix (ParAccel)
Flat files	Pivotal Greenplum
SharePoint	Pivotal Hawq
ftp locations	Amazon Redshift
Hadoop (hdfs)	

HVR's MFT capabilities

Setting up HVR is done from HVR's easy to use GUI. The GUI exposes all options to the user in easy dialogs. The GUI also shows the commands needed to perform the same actions on the command line for scripting as well. Finally, the GUI is directly linked to HVR's help system through context-sensitive help.

The first step is to define physical locations for files. These locations can be local or remote. Local locations are just locations on your file system (a directory). Remote locations exists in 2 flavors: connecting to a remote HVR agent accessing its file system or connect through an external protocol. The former uses HVR's built in protocol, ensuring optimal performance and reliability

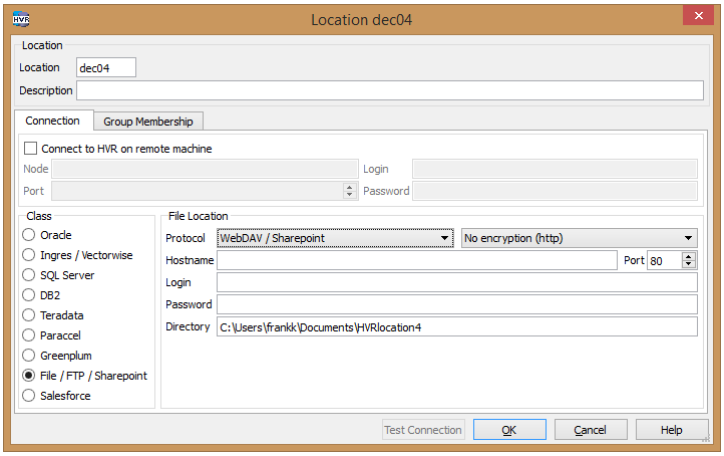


Figure 1 Setting up a remote file location

However, in certain cases, only external protocols are available. HVR allows you to connect through webdav or (s)ftp(s). You'll have to provide the credentials and set the necessary options (encryption, port, etc.). The advantage of using these protocols with HVR rather than in scripting is the monitoring and managing capabilities HVR adds.

The next step is setting up a transfer, in HVR terms a "Channel". A channel can contain various types of locations (database or file). In the case of MFT, there are file locations involved. If there is no data structure in the channel definition (e.g. a table layout) then HVR handles files as BLOBs. BLOBs can be of any format and transported anywhere, but they cannot be transformed. If the user adds structure to a channel by defining table structures then HVR treats files as containing this structure (in XML or CSV format). In this case HVR can manipulate the data in the files.

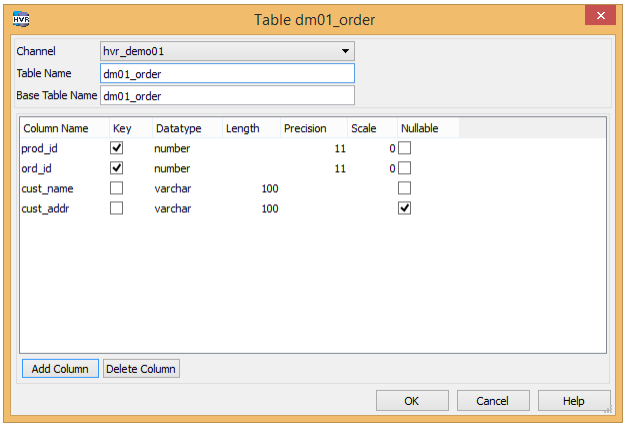


Figure 2 Defining table structure defines file structure as well

Should you wish to handle structured and unstructured data together, you can create 2 separate channels sharing the locations, one with table structures defined, one without.

In order to define what has to be done in a channel, you have to define “Actions” in the channel. Every channel needs an action for the source and an action for the target: the “FileCapture” and “FileIntegrate” respectively.

Defining Location behavior

The Action ”FileCapture” is used to instruct HVR to capture files from a file location and to control its behavior.

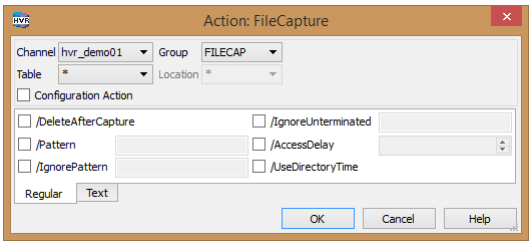


Figure 3 Action FileCapture

Defining the action is sufficient to start replication. You can however taylor it to your needs by setting options. Use the /DeleteAfterCapture option to move files instead of copying them. The /Pattern and /IgnorePattern options control what files are picked up and ignored by HVR. For example, pickup all files called *.xml and ignore all files with tmp in it (*tmp*). More powerful expressions are possible.

There are multiple ways to control the handshake between the process that creates the file and HVR picking it up. Ideally the creation of a file is an atomic process, e.g. by moving it to the capture location when the file is complete. Alternatively, the /IngoreUnterminated option can be used for HVR to wait until a termination expression is written to a file (e.g. EOF). /AccessDelay forces HVR to wait for a certain amount of time before processing a file. This can be used when writing the file is guaranteed to finish in a certain amount of time. The /UseDirectoryTime option can be used in conjunction with touching the directory date when done writing.

For the target file locations, the “FileIntegrate” action has to be set:

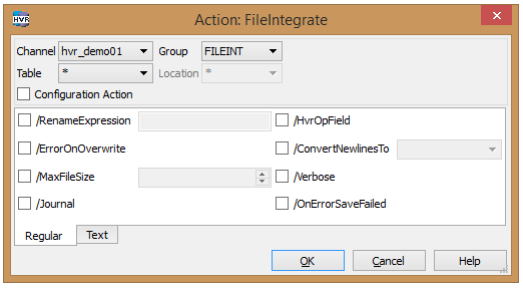


Figure 4 Action FileIntegrate

Again, setting it is enough to commence transfer of files. However, several options can be controlled.

/ErrorOnOverwrite controls whether overwrites are allowed or not. Overwrites would normally only happen when the source file is being altered and HVR has to retransfer it. The parameter /RenameExpression allows the user to perform rename operations with regular expressions. You can also add variables like a timestamp to filenames (which can be used to prevent overwrites from happening). /MaxFileSize will split files accordingly, it is normally only used on structured files containing data. /Journal enables journaling, /HVROpField adds an operation field, enabling deletes to be written as well. This is usefull in database / file combinations. /ConvertNewLines controls Dos/Unix newline conversion and /OnErrorSaveFailed will save the error induced files to a different location, making the transfer more resilient by not aborting it.

HVR's advanced options

When more is needed than a 1-1 file transfer, HVR's advanced actions are to be called in.

To transform files, the action "Transform" is available within HVR.

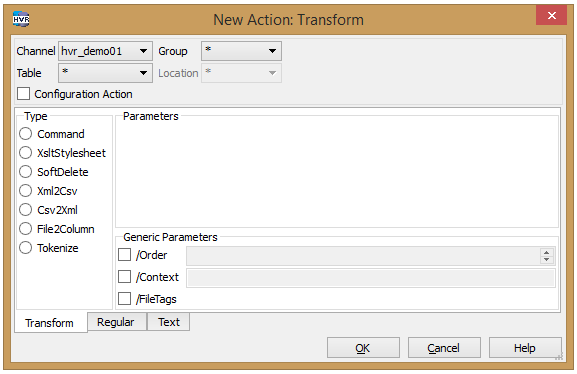


Figure 5 Action Transform

In this action, it is possible to add transformations. HVR supports a number of different transformation mechanisms. HVR has some transformations buit-in like:

- Softdeletes (transforming a deletion indication to items representing a a delete
- XML <=> CSV
- Tokenize (call an external token service to encrypt a value)
- Fike2Column (load a file into a database column)

On top of that HVR allows you to add an XSLT style sheet performing the transformation and finally allows you to write you own conversion script and control it from HVR. This also allows you to interface with another application directly.

A popular use case for external transformations is the use of HVR in conjunction with an ETL tool. For these scenarios it often makes sense to use the /HVROpField parameter in conjunction with the Command option in anTransformation action.

Using multiple transformation actions is also possible. You can define the Transform action multiple times and can control their sequence with the /order option.

For advanced option on capture and integrate location, an action “LocationProperties” can be used in conjunction with FileCapture and/or FileIntegrate

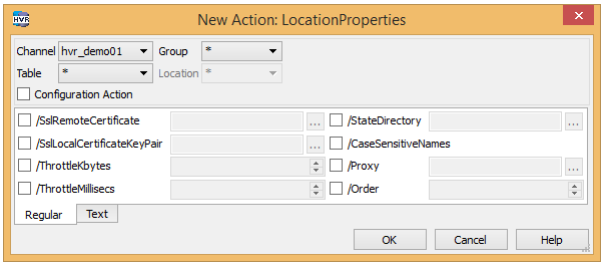


Figure 6 Defining LocationProperties

It enables you to encrypt the connection to the location (/SslCertificate), connect through a proxy (/Proxy) and throttle the used bandwidth (/ThrottleKbytes and /ThrottleMillisecs).

For read only locations, the creation of temporary files is impossible. However, HVR normally creates a subdirectory “State” for all locations into or out of which HVR writes or reads. This location can be overridden in the action “LocationProperties” with the /StateDirectory parameter.