DELPHIX

Delphix DB2 Database Guide

June 2018

DB2 Environments and Data Sources

The following pages will walk users through engine specific requirements and configurations of source and target environments in order to complete the following:

- Add and manage environments to the Delphix Engine
- Link and manage dSources
- Provision virtual databases or virtual files
- Manage virtualization environments

DB2 on Delphix: An Overview

Introduction to DB2

DB2 for Linux, UNIX and Windows is a database server product developed by IBM. Sometimes called DB2 LUW for brevity, it is part of the DB2 family of database products. DB2 LUW is the "Common Server" product member of the DB2 family, designed to run on most popular operating systems. By contrast, all other DB2 products are specific to a single platform.

DB2 LUW was initially called DB2 Universal Database (UDB), but over time IBM marketing started to use the same term for other database products, notably mainframe (z-Series) DB2. Thus the DB2 for Linux, UNIX and Windows moniker became necessary to distinguish the common server DB2 LUW product from single-platform DB2 products.

The current DB2 LUW product runs on multiple Linux and UNIX distributions, such as Red Hat Linux, SUSE Linux, AIX, HP/UX, and Solaris, and most Windows systems. Multiple editions are marketed for different sizes of organization and uses. The same code base is also marketed without the DB2 name as IBM InfoSphere Warehouse edition.

The version numbers in DB2 are non-sequential with v10.1 and 10.5 being the two most recent releases. Specifics of DB2 versions and platforms supported on Delphix are located in the DB2 Compatibility Matrix.

DB2 Authentication

Authentication is the process of validating a supplied user ID and password using a security mechanism. User and group authentication is managed in a facility external to DB2 LUW, such as the operating system, a domain controller, or a Kerberos security system. This is different from other database management systems (DBMSs), such as Oracle and SQL Server, where user accounts may be defined and authenticated in the database itself, as well as in an external facility such as the operating system.

Any time a user ID and password is explicitly provided to DB2 LUW as part of an instance attachment or database connection request, DB2 attempts to authenticate that user ID and password using this external security facility. If no user ID or password is provided with the request, DB2 implicitly uses the user ID and password that were used to login to the workstation where the request originated. More information on DB2 authentication and authorization is available via IBM documentation.

Delphix DB2 Authentication
Delphix for DB2 requires that that the staging and target hosts must already have the necessary users and authentication systems created/installed on them. Delphix will neither create users nor change database passwords as part of the provisioning process.

DB2 Database Level Support

DB2 database level support contains:

- Support of multiple database linking in a Single Instance, which allows Delphix Engine users to utilise an available instance on the target more efficiently.
- Supports the use of customer supplied directory for DE Toolkit, DB2 Mount Points, and DB2 Delphix files/logs. Customers can now easily manage all of their content in once place.
- Supports BLU feature which is an in-memory concept of DB2.
- Supports Kerberos environments.
- Support for VDB at same OS version level or one level higher than Staging/Production.
- Intelligent handling of HADR log-gaps.

High Availability Disaster Recovery (HADR)

The HADR feature of IBM DB2 provides a high availability solution for both partial and complete site failures. It protects against data loss by replicating data changes from a source database, called the primary, to one or more target databases, called the standby.

Delphix HADR
HADR replication takes place at a database level, not at the instance level. Therefore, a standby instance can have multiple databases from multiple different primary servers/instances on it. If the instance ID on the Delphix standby is NOT the same as the instance ID on the primary, the Delphix standby instance ID MUST have database permissions **secadm** and **dbadm** granted to it on the primary database. These permissions, and all HADR settings, must be implemented on the primary database BEFORE you take the backup on the primary database.

Log Transmitting

All changes that take place at the primary database server are written into log files. The individual log records within the log files are then transmitted to the secondary database server, where the recorded changes are replayed to the local copy of the database. This procedure ensures that the primary and the secondary database servers are in a synchronized state. Using two dedicated TCP/IP communication ports and a heartbeat, the primary and the standby databases track where they are processing currently, the current state of replication, and whether the standby database is up-to-date with the status of the primary database. When a log record is "closed" (still in memory, but has yet to be written to disk on the primary), it is immediately transmitted to the HADR standby database(s). Transmission of the logs to the standbys may also be time-delayed.

Multiple Standby

Beginning in DB2 v10.1,the HADR feature supports multiple standby databases. This enables an advanced topology where you can deploy HADR in multiple standby mode with up to three standby databases for a single primary. One of the databases is designated as the principal HADR standby database, with the others termed as auxiliary HADR standby databases. As with the standard HADR deployment, both types of HADR standbys are synchronized with the HADR primary database through a direct TCP/IP connection. Furthermore, both types support the reads on standby feature and can be configured for time-delayed log replay. It is possible to issue a forced or unforced takeover on any standby, including the delphix auxiliary standby. However, you should never use the Delphix auxiliary standby as a primary, because this will impact Delphix performance.

Delphix HADR Synchronization

The Delphix for DB2 uses the HADR capability of DB2 to synchronize data from a production DB2 database into a Delphix-controlled DB2 "standby" server. By using this mature and existing DB2 capability, the Delphix Engine is able to ingest data and keep the standby server in sync with only a minimal impact on production. The HADR connection is configured to Super-Asynchronous (SUPERASYNC) mode where log writes are considered successfully transmitted when the log records are sent from the primary database. Because the primary database does not wait for acknowledgements from the standby database, there is no delay on the primary and transactions are considered committed regardless of the state of the replication of that transaction. For further information on Delphix synchronization, see Linking a dSource from a DB2 Database: An Overview.

DB2 Support and Requirements

These topics describe specific requirements for DB2 environments, such as user privileges and the supported operating systems and database versions.

- DB2 Compatibility Matrix
- Network and Connectivity Requirements for DB2 Environments
- Requirements for DB2 Hosts and Databases
- Sudo File Configuration Examples for DB2 Environments
- Sudo Privilege Requirements for DB2 Environments

DB2 Compatibility Matrix

This topic describes the DB2 (DBMS) versions that are supported by Delphix, as well as the compatible operating systems (OS), for use on target and source environments.

Source, Staging and Target OS and DBMS Compatibility

The source, staging, and target hosts must all be running the same DBMS/Operating System combination (for example, DB2 10.5.4 on RHEL 6.5) in order to successfully provision a VDB to the target.

Different fix packs of same DB2 versions across source, staging, and target are supported with the following exceptions :

1. Source on 10.0.5.x, Staging on 10.0.5.4, where x is greater than 4. This can be fixed by validating the invalid packages on Staging.
2. Staging on 10.0.5.x, Target on 10.0.5.4, where x is higher than 4
3. Source on 11.1.2.2, Staging on 11.1.1.1
4. Staging on 11.1.2.2, Target on 11.1.1.1
5. Source on 11.1.3.3, Staging on 11.1.2.2
6. Staging on 11.1.3.3, Target on 11.1.2.2

Supported DBMS Versions

- DB2 Enterprise Server Edition 10.1
- DB2 Advanced Enterprise Server Edition 10.1
- DB2 Enterprise Server Edition 10.5
- DB2 Advanced Enterprise Server Edition 10.5
- DB2 Enterprise Server Edition 11.1
- DB2 Advanced Enterprise Server Edition 11.1
- DB2 Developer Edition 11.1 (on RHEL 6.7)

Supported Operating Systems

- Red Hat Enterprise Linux (RHEL)
- Advanced Interactive eXecutive (AIX)
- Toolkit/Delphix Engine Compatibility

1. ESE: Enterprise Server Edition
2. AESE: Advanced Enterprise Server Edition
3. As of Delphix Engine 5.1.6 - For the supported DB2 versions, Delphix supports the corresponding DB2 Developer edition where the vendor, IBM, supports it
4. 64-bit OS support only

Red Hat Enterprise Linux (RHEL)

Supported DBMS Version	ESE 10.5	AESE 10.5	ESE 11.1	AESE 11.1
Supported OS Version				
RHEL 6.0	Not Supported	Not Supported	Not Supported	Not Supported
RHEL 6.1	Not Supported	Not Supported	Not Supported	Not Supported
RHEL 6.2	Not Supported	Not Supported	Not Supported	Not Supported
RHEL 6.3	Not Supported	Not Supported	Not Supported	Not Supported
RHEL 6.4	Supported	Supported	Not Supported	Not Supported
RHEL 6.5	Supported	Supported	Not Supported	Not Supported
RHEL 6.6	Supported	Supported	Not Supported	Not Supported
RHEL 6.7	Supported	Supported	Supported	Supported
RHEL 6.8	Supported	Supported	Supported	Supported
RHEL 6.9	Supported	Supported	Supported	Supported
RHEL 7.0	Supported	Supported	Supported	Supported
RHEL 7.1	Supported	Supported	Supported	Supported

	RHEL 7.2	Supported	Supported	Supported	Supported
	RHEL 7.3	Supported	Supported	Supported	Supported
	RHEL 7.4	Supported	Supported	Supported	Supported

Advanced Interactive eXecutive (AIX)

Supported DBMS Version		**ESE 10.5**	**AESE 10.5**	**ESE 11.1**	**AESE 11.1**
	Supported OS Version				
	AIX 6.1	Not Supported	Not Supported	**N/A**	**N/A**
	AIX 7.1	Supported	Supported	Supported	Supported
	AIX 7.2	Supported	Supported	Supported	Supported

Toolkit/Delphix Engine Compatibility

Toolkits should be installed on compatible Delphix Engines per the table below:

Delphix Engine	**DB2_2.0.1**	**DB2_2.1.0**	**DB2_2.2.1**	**DB2_2.3.0**
5.1.x	No	No	No	No
5.2.2	Yes	No	No	No
5.2.3	No	Yes	Yes	Yes
5.2.4	No	Yes	Yes	Yes
5.2.5	No	Yes	Yes	Yes

Unsupported DB2 Versions and Features

- DB2 9.7 and below
- DB2 Database Partitioning Feature (DPF)
- DB2 pureScale
- DMS Raw Devices

Network and Connectivity Requirements for DB2 Environments

This topic outlines the network and connectivity requirements for the Delphix Engine and DB2 standby and target environments.

- Port Allocations Specific to DB2
    - Inbound to the Delphix Engine Port Allocation
    - Outbound from a Standby or Target Environment Port Allocation
    - Inbound to a Standby or Target Environment Port Allocation
    - HADR Service Ports
- General Outbound from the Delphix Engine Port Allocation
- General Inbound to the Delphix Engine Port Allocation
- Firewalls and Intrusion Detection Systems (IDS)
- AppData Port Requirements

Port Allocations Specific to DB2

The Delphix Engine makes use of the following network ports for DB2 standby and target:

Inbound to the Delphix Engine Port Allocation

Protocol	Port Number	Use
TCP/UDP	111	Remote Procedure Call (RPC) port mapper used for NFS mounts Note: RPC calls in NFS are used to establish additional ports, in the high range 32768-65535, for supporting services. Some firewalls interpret RPC traffic and open these ports automatically. Some do not.
TCP	1110	NFS Server daemon status and NFS server daemon keep-alive (client info)
TCP/UDP	2049	NFS Server daemon from vFiles to the Delphix Engine
TCP	4045	NFS lock daemon/manager
UDP	33434 - 33464	Traceroute from standby and target hosts to the Delphix Engine (optional)
UDP/TCP	32768 - 65535	NFS mountd and status services, which run on a random high port. Necessary when a firewall does not dynamically open ports.

Outbound from a Standby or Target Environment Port Allocation

Protocol	Port Numbers	Use
TCP	873	Rsync connections used during V2P.
TCP	8415	DSP connections used for monitoring and script management. Typically DSP runs on port 8415.

Inbound to a Standby or Target Environment Port Allocation

Protocol	Port Numbers	Use
TCP	22	SSH connections to the target environment

HADR Service Ports

The HADR ports set for HADR_LOCAL_SVC and HADR_REMOTE_SVC on the DB2 Master and Standby hosts. The specific ports used at the customers' discretion and need to be specified during the linking process. It is highly recommended that this ports also be defined in the /etc/services file to ensure that they are only used by DB2 for the specified databases.

General Outbound from the Delphix Engine Port Allocation

Protocol	**Port Numbers**	**Use**
TCP	25	Connection to a local SMTP server for sending email
TCP/UDP	53	Connections to local DNS servers
UDP	123	Connection to an NTP server
UDP	162	Sending SNMP TRAP messages to an SNMP Manager
TCP	443	HTTPS connections from the Delphix Engine to the Delphix Support upload server
TCP/UDP	636	Secure connections to an LDAP server
TCP	8415	Connections to a Delphix replication target. See Configuring Replication.
TCP	50001	Connections to source and target environments for network performance tests via the Delphix command line interface (CLI). See Network Performance Tool.

General Inbound to the Delphix Engine Port Allocation

Protocol	Port Number	Use
TCP	22	SSH connections to the Delphix Engine
TCP	80	HTTP connections to the Delphix GUI
UDP	161	Messages from an SNMP Manager to the Delphix Engine
TCP	443	HTTPS connections to the Delphix Management Application
TCP	8415	Delphix Session Protocol connections from all DSP-based network services including Replication, SnapSync for Oracle, V2P, and the Delphix Connector.
TCP	50001	Connections from source and target environments for network performance tests via the Delphix CLI. See Network Performance Tool.
TCP/UDP	32768 - 65535	Required for NFS mountd and status services from target environment only if the firewall between Delphix and the target environment does not dynamically open ports. **Note:** If no firewall exists between Delphix and the target environment, or the target environment dynamically opens ports, this port range is not explicitly required.

Firewalls and Intrusion Detection Systems (IDS)

Production databases on source environments (for dSources) are often separated from the non-production environment by firewalls. Firewalls can add milliseconds to the latency between servers. Accordingly, for best performance, there should be no firewalls between the Delphix Engine and the virtual database (VDB) target environments. If the Delphix Engine is separated from a source environment by a firewall, the firewall must be configured to permit network connections between the Delphix Engine and the source environments for the application protocols (ports) listed above.

Intrusion detection systems (IDSs) should also be made permissive to the Delphix Engine deployment. IDSs should be made aware of the anticipated high volumes of data transfer between dSources and the Delphix Engine.

AppData Port Requirements

The use of AppData requires the following ports/protocols.
Two important notes about these specifications:

1. The next release of the Delphix Engine will significantly augment the port/protocol utilization of AppData. The upcoming-only requirements have been marked with a *.
2. AppData V2P uses RSYNC to export to the target. RSYNC between the target and Delphix Engine is not required for general virtualization usage. The V2P-only requirements have been marked with a ^.

From Source to Delphix Engine	From Delphix Engine to Source	From Target to Delphix Engine	From Delphix Engine to Target
RSYNC (TCP Port 873)	RSYNC (TCP Port 873)	DSP (Default TCP Port 8415)	DSP (Default TCP Port 8415)
DSP (Default TCP Port 8415)	SSH (TCP Port 22)	NFS	SSH (TCP Port 22)
*NFS	DSP (Default TCP Port 8415)	^RSYNC (TCP Port 873)	^RSYNC (TCP Port 873)

Requirements for DB2 Hosts and Databases

DB2 hosts are servers that have DB2 binaries installed and have DB2 instances created on them. The hosts that contain the data that we wish to ingest are referred to as the source environment. Hosts with empty instances (no dbs in instance) are used as either staging or target hosts. This topic describes the requirements for creating connections between the Delphix Engine and DB2 hosts and instances.

Requirements for DB2 Source Hosts and Instances

Each DB2 Source host (master) must meet these requirements:

- IBM DB2 installed and instance created on the machine
- HADR settings for each database to be used with the standby server should be preset before the linking process begins as described in Linking a DB2 dSource

Requirements for DB2 Staging and Target Hosts and Instances

- The staging environment that the Delphix Engine uses must have access to an existing full backup of the source database on disk to create the first full copy. Delphix recommends using compressed backups as that will reduce storage needs and speed up ingest.
- The staging and target DB2 instances that you wish to use must already exist on the host and contain no existing databases.
- The available instances on each host can be verified by going to the databases tab for the environment in question.

Additional Environment Requirements

- There must be an operating system user (**delphix_os**) with these privileges:
  - Ability to login to the target environment via SSH
  - Ability to run `mount`, `umount`, `mkdir`, and `rmdir` as a super-user. If the target host is an AIX system, permission to run the nfso command as a super-user. See Sudo Privilege Requirements for DB2 Environments for further explanation of the commands and Sudo File Configuration Examples for DB2 Environments for examples of the `/etc/sudoers` file on different operating systems.
- There must be a directory on the staging and target environment where you can install the Delphix Engine Toolkit – for example, `/var/opt/delphix/toolkit` .

  - The **delphix_os** user must own the directory.
  - The directory must have permissions -rwxrwx--- (0770), but you can also use more permissive settings.
  - If delphix os user and instance users (responsible for the Delphix Engine operations such as linking and provisioning) are not sharing any common group, then toolkit directory must have -rwxrwxrwx (0777) permissions.
  - The **delphix_os** user must have read and execute permissions on each directory in the path leading to the toolkit directory. For example, when the toolkit is stored in `/var/opt/delphix/toolkit`, the permissions on `/var`, `/var/opt`, and `/var/opt/delphix` should allow read and execute for "others," such as -rwxr-xr-x.
  - The directory should have 1.5GB of available storage: 400MB for the toolkit and 400MB for the set of logs generated by each DB2 instance that runs on the host.
  - In DB2 Toolkit: toolkit directory space will be used as the base location for the mount point.
- The Delphix Engine must be able to initiate an SSH connection to the target environment
- NFS client services must be running on the target environment

Instance User Requirements

- The instance owner of each instance you wish to use within a staging or a target host must be added as an environment user within the Delphix engine. See Managing DB2 Users and Instance Owners.
- For HADR synced dSources the staging instance owner must be able to "read" the ingested database contents as Delphix will check the validity of the database by querying tables before each dSource snapshot.

Database Container Requirements

- All DB2 database containers types are fully supported with the exception of DB2 raw containers. NOTE: If a container is added or deleted, the dSource will have to be resynced.

Instance level configuration values such as the bufferpool value will need to be managed by the customer independent of Delphix. The instances used for staging and target environments must be compatible with the source DB2 instance. The Delphix DB2 DB Level toolkit supports managing dSources with database level granularity.

Related Links

- DB2 Compatibility Matrix
- Setting Up DB2 Environments: An Overview

Sudo Privilege Requirements for DB2 Environments

This topic describes the rationale behind specific `sudo` privilege requirements for virtualizing DB2 Databases.

Privilege	Sources	Targets and Staging	Rationale
`mkdir/rmdir`	Not Required	**Required**	Delphix dynamically makes and removes directories under the provisioning directory during VDB operations. This privilege is optional, provided the provisioning directory permissions allow the delphix os user to make and remove directories.
`mount/umount`	Not Required	**Required**	Delphix dynamically mounts and unmounts directories under the provisioning directory during VDB operations. This privilege is required because `mount` and `umount` are typically reserved for superuser.

It is required to specify the NOPASSWD qualifier within the "sudo" configuration file, as shown here: Sudo File Configuration Examples for DB2 Environments. This ensures that the "sudo" command does not demand the entry of a password, even for the "display permissions" (i.e. "sudo -l") command.

Delphix issues "sudo -l" in some scripts to detect if the operating system user has the correct sudo privileges. If it is unable to execute this command, some actions may fail and Delphix will raise an alert suggesting it does not have the correct sudo permissions. Restricting the execution of "sudo -l" by setting “listpw=always” in the “/etc/sudoers” file when the Delphix operating system user is configured to use public key authentication will cause the Delphix operating system user to be prompted for a password which will fail certain Delphix actions. Use a less restrictive setting for listpw than "always" when the Delphix operating system user is using public key authentication.

Related Links

- Requirements for DB2 Hosts and Databases
- Sudo File Configuration Examples for DB2 Environments

Sudo File Configuration Examples for DB2 Environments

This topic provides sample `sudo` file privilege configurations for using the Delphix Engine with various operating systems and the Oracle RDBMS.

Configuring sudo Access on Linux for DB2 Source and Target Environments

On a Linux target, sudo access to `mount`, `umount`, `mkdir`, and `rmdir` is required.

Example: Linux /etc/sudoers file for a Delphix Target for DB2

```
Defaults:delphix_os !requiretty
delphix_os ALL=NOPASSWD: \
/bin/mount, /bin/umount, /bin/mkdir, /bin/rmdir
```

Examples of Limiting sudo Access for the Delphix OS User

In situations where security requirements prohibit giving the Delphix user root privileges to mount, unmount, make directory, and remove directory on the global level, it is possible to configure the `sudoers` file to provide these privileges only on specific mount points or from specific Delphix Engines, as shown in these two examples.

The Delphix Engines tests its ability to run the `mount` command using `sudo` on the target environment by issuing the `sudo mount` command with no arguments. Many of the examples shown in this topic do not allow that. This causes a warning during environment discovery and monitoring, but otherwise does not cause a problem. If your VDB operations succeed, it is safe to Ignore this warning.

However, some users configure the security on the target environments to monitor `sudo` failures and lock out the offending account after some threshold. In those situations, the failure of the sudo commands might cause the **delphix_os** account to become locked. One work-around for this situation is to increase the threshold for locking out the user account. Another option is to modify `/etc/sudoers` to permit the **delphix_os** user to run `mkdir, rmdir, umount` and `mount` command without parameters.

Example 1

This example restricts the **delphix_os** user's use of `sudo` privileges to the directory `/db2`.

Note that wildcards are allowed for the options on `mount` and `umount` because those commands expect a fixed number of arguments after the options. The option wildcard on the `mount` command also makes it possible to specify the file-system being mounted from the Delphix Engine.

However, wildcards are not acceptable on `mkdir` and `rmdir` because they can have any number of arguments after the options. For those commands, you must specify the exact options (`-p`, `-p -m 755`) used by the Delphix Engine.

Delphix requires `umount -lf` for emergency force unmounts on Linux. For other Unix OSes, Delphix requires `umount -f`.

Example /etc/sudoers File Configuration on the Target Environment for sudo Privileges on the VDB Mount Directory Only (Linux OS)

```
Defaults:delphix_os !requiretty
delphix_os ALL=(root) NOPASSWD: \
/bin/mount  *        /db2/*, \
/bin/umount *        /db2/*, \
/bin/umount          /db2/*, \
/bin/umount -lf      /db2/*, \
/bin/mkdir -p        /db2/*, \
/bin/mkdir -p -m 755 /db2/*, \
/bin/mkdir           /db2/*, \
/bin/rmdir           /db2/*
```

Example 2

This example restricts the **delphix_os** user's use of `sudo` privileges to the directory `/db2`, restricts the mount commands to a specific Delphix Engine hostname and IP, and does not allow user-specified options for the `umount` command.

Delphix requires `umount -lf` for emergency force unmounts on Linux. For other Unix OSes, Delphix requires `umount -f`.

This configuration is more secure, but there is a tradeoff with deployment simplicity. This approach would require a different sudo configuration for targets configured for different Delphix Engines.

A Second Example of Configuring the /etc/sudoers File on the Target Environment for Privileges on the VDB Mount Directory Only, and Allows Mounting Only from a Single Server (Linux OS)

```
Defaults:delphix_os !requiretty
delphix_os ALL=(root) NOPASSWD: \
/bin/mount           <delphix-server-name>* /db2/*, \
/bin/mount *         <delphix-server-name>* /db2/*, \
/bin/mount           <delphix-server-ip>*   /db2/*, \
/bin/mount *         <delphix-server-ip>*   /db2/*, \
/bin/mount "", \
/bin/umount          /db2/*, \
/bin/umount *        /db2/*, \
/bin/umount -lf      /db2/*, \
/bin/mkdir [*]       /db2/*, \
/bin/mkdir           /db2/*, \
/bin/mkdir -p        /db2/*, \
/bin/mkdir -p -m 755 /db2/*, \
/bin/rmdir           /db2/*
```

Related Links

- Sudo Privilege Requirements for DB2 Environments
- Requirements for DB2 Hosts and Databases

Managing DB2 Environments

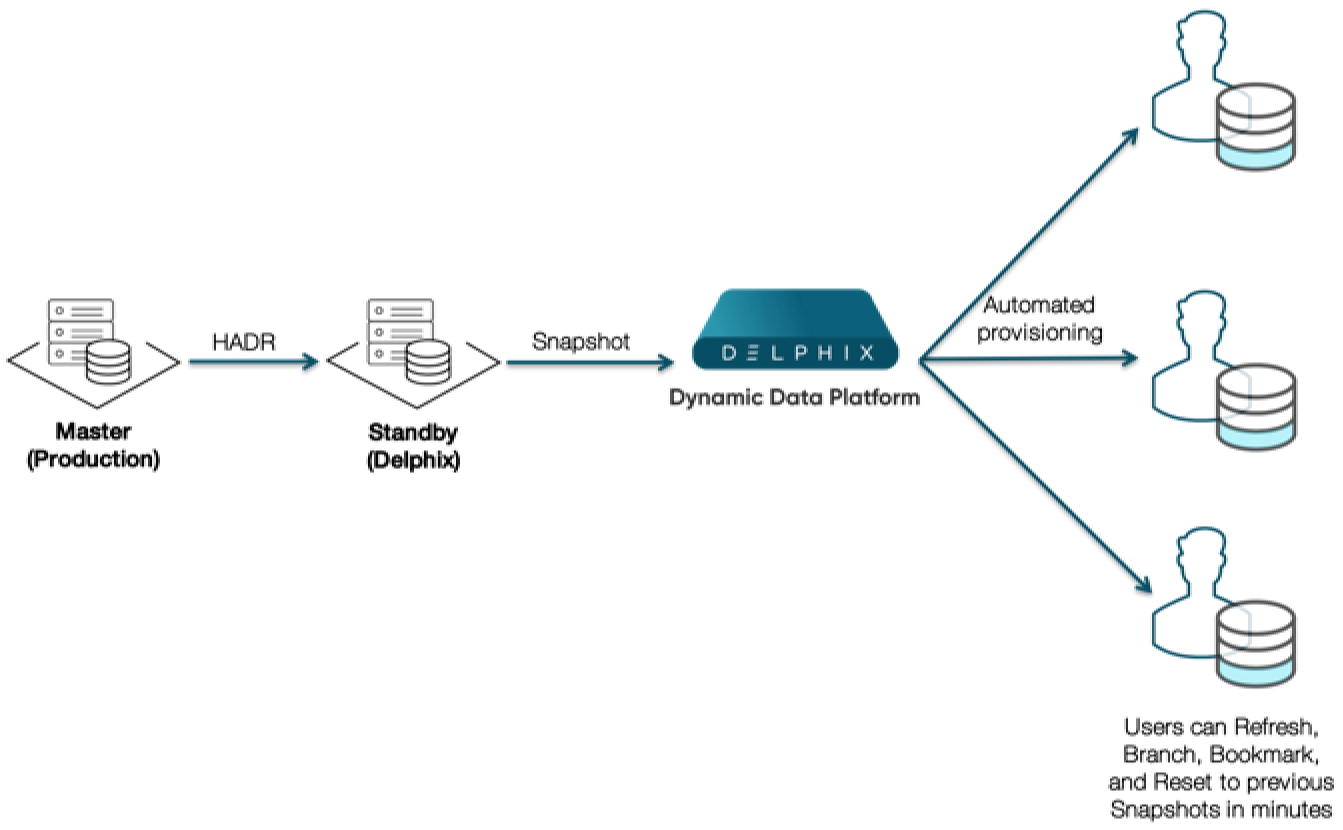
These topics describe special tasks and concepts for working with DB2 instances and databases

- Setting Up DB2 Environments: An Overview
- Editing DB2 Environment Attributes
- Managing DB2 Instances
- Managing DB2 Users and Instance Owners
- Deleting a DB2 Environment
- Refreshing a DB2 Environment

Setting Up DB2 Environments: An Overview

This topic describes the high-level process for adding DB2 environments, linking DB2 instances to the Delphix Engine, and provisioning virtual databases.

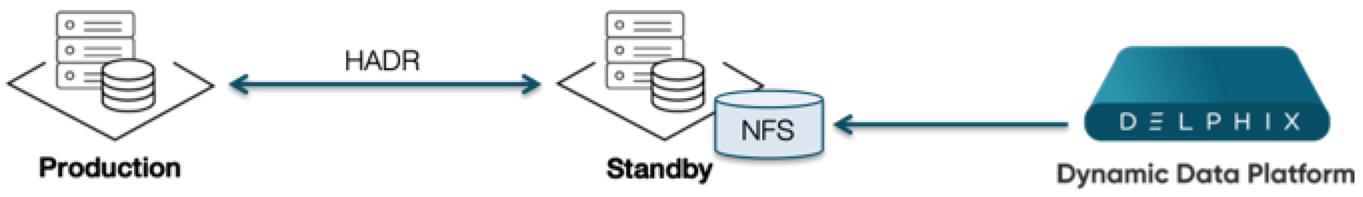
Delphix for DB2 Architecture



Delphix uses a Standby server model along with DB2s High Availability Disaster Recovery feature to ingest data and stay in sync with the source database. The standby server is then snapshotted by the Delphix Engine and the snapshots can be provisioned out to one or more target servers.

> The snapshot and provision process occurs on the instance level, all databases that exist on the standby server will be provisioned out to the target machines. Similarly, actions such as bookmark, rewind, and refresh will simultaneously apply to all the databases in the instance.

Block Diagram of Linking Architecture Between DB2 Environments and the Delphix Engine



The linking process sets up a database inside an instance on the standby server and uses this as a HADR standby for the primary database. The staging instance must have access to a recent backup copy of the database that will be used to restore the initial copy of the dSource. Once the restoration process is complete, Delphix will issue the HADR standby commands for the given database and ensure that the health of the HADR connection stays within the acceptable threshold values.

DB2 Staging Instance Set Up

Database level feature enables you to have 1-many mappings between instance and databases. It is also possible to set up databases from multiple primary hosts to use the same standby instance.

The choice of databases on the staging server should also take into account the expected network traffic that HADR will create between the source and staging environments.

Related Links

- DB2 Support and Requirements

Editing DB2 Environment Attributes

This topic describes how to edit attributes of an environment such as name, host address, ssh port, or toolkit path, as well as describing more advanced attributes for specific data platforms.

Procedure

1. Login to the **Delphix Management** application with Delphix Admin credentials or as the owner of an environment.
2. Click **Manage**.
3. Select **Environments**.
4. In the **Environments** panel, click the name of an environment to view its attributes.
5. Next to **Attributes**, click the **Pencil** icon to edit an attribute.
6. Click the **Check** icon to save your edits.

Common Editable Attributes

Attribute	Description
Environment Users	The users for that environment. These are the users who have permission to ssh into an environment or access the environment through the Delphix Connector. For more information on the environment user requirements, see the **Requirements** topics for specific data platforms.
Host Address	The IP address of the environment host.
SSH Port	The SSH port number used for the environment.
Toolkit Path	The toolkit path for the environment. In DB2 DB level toolkit, we are using toolkit path for logs, mount points and keeping lib files. The structure will be as below:- ◦ For Mount points : <toolkit dir>/DB2/mnts/<env user>/<DB Name>/<env user> ◦ For logs : <toolkit dir>/DB2/logs/<env user> ◦ For code :<toolkit dir>/DB2/code/<env user>
Notes	Any other information you want to add about the environment

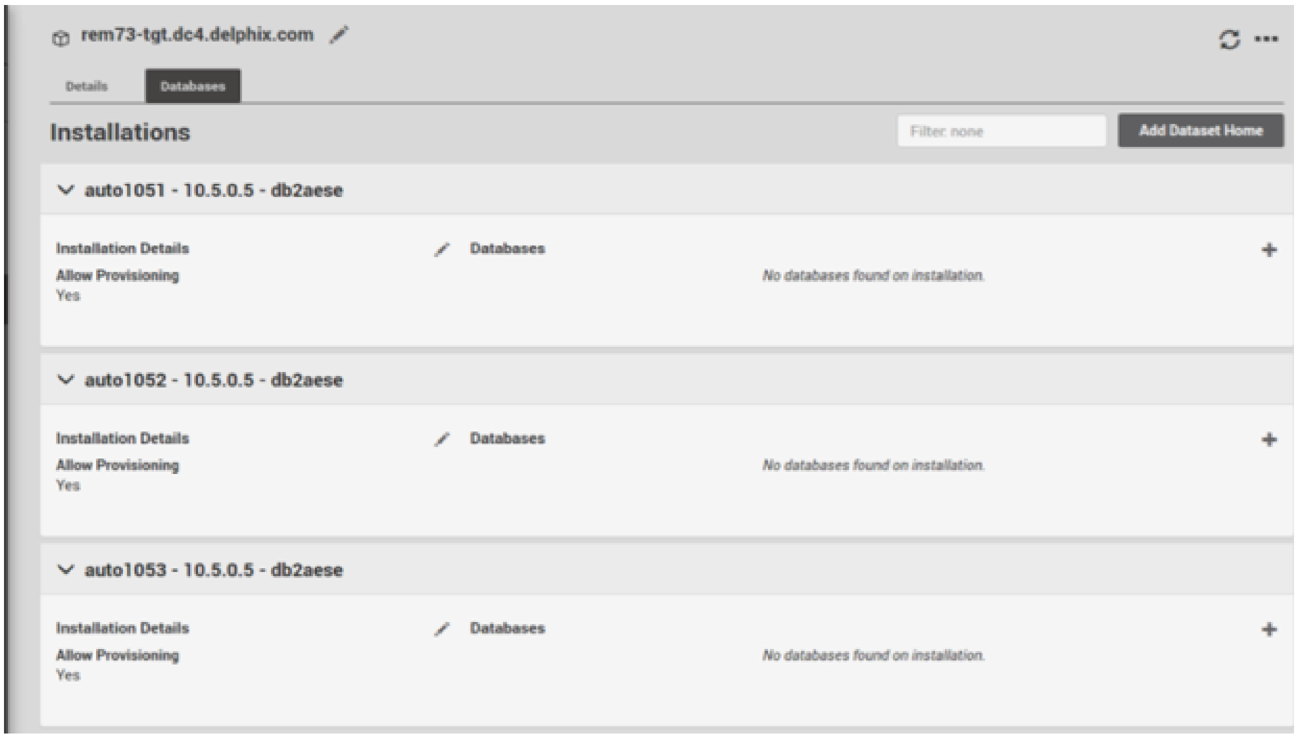
Managing DB2 Instances

This topic explains how to manage DB2 instances on a host.

When you add an environment with the Delphix Management Application, all existing DB2 instances on the host are automatically discovered by Delphix. A list of all instances and databases are available to Delphix based on the environment discovery process.

View Instances

1. Login to the **Delphix Management** application with Delphix Admin credentials.
2. Click **Manage**.
3. Select **Environments**.
4. In the **Environments** panel, click on the **name of the environment** to you want to refresh.
5. Select the **Databases** tab to see a list of all DB2 instances found in the environment.



Installations

auto1051 - 10.5.0.5 - db2aese

Installation Details
Allow Provisioning
Yes
Databases
No databases found on installation.

auto1052 - 10.5.0.5 - db2aese

Installation Details
Allow Provisioning
Yes
Databases
No databases found on installation.

auto1053 - 10.5.0.5 - db2aese

Installation Details
Allow Provisioning
Yes
Databases
No databases found on installation.

Updating Instances on a Host

If you add or remove any DB2 instances from a host you must refresh the environment using the steps listed in Refreshing a DB2 Environment to update the list of instances available to Delphix. Additionally, the environment users must be updated to match the DB2 instance owners as detailed in Managing DB2 Users and Instance Owners.

Managing DB2 Users and Instance Owners

All DB2 Delphix operations such as linking, snapshot and provisioning always occur on the instance level. As a result we require that any DB2 instance that you wish to use either as Staging or Target must have its associated instance owner added as an environment user to the host within Delphix. Please perform the following steps to add each DB2 instance owner you to use on the host.

It is important to note that Delphix will not create new instances on the host and can only use existing DB2 instances that finds on the host machine. For more information on instance management refer to Managing DB2 Instances.

Adding Environment Users

Procedure

1. Login to the Delphix Management application using Delphix admin credentials.
2. Click **Manage**.
3. Select **Environments**.
4. Click on the existing environment name you want to modify and open the environment information screen.
5. In the Details tab, click the **Plus** icon located next to Environment users.
6. Enter the Username and Password for the OS user in that environment.

If you want to use public key encryption for logging into your Unix-based environment:

a. Select **Public Key** for the **Login Type.**
b. Click **View Public Key**.
c. Copy the public key that is displayed, and append it to the end of your `~/.ssh/authorized_keys` file. If this file does not exist, you will need to create it.
    i. Run `chmod 600 authorized_keys` to enable read and write privileges for your user.
    ii. Run `chmod 755 ~` to make your home directory writable only by your user.

The public key needs to be added only once per user and per environment.

7. Click the **Check** icon to save the new user.
8. To change the primary user for this environment, click the **Pencil** icon next to Environment Users. Only the primary user will be used for environment discovery.
9. To delete a user, click the **Trash** icon next to their username.

Deleting a DB2 Environment

This topic describes how to delete an environment. Deleting an environment only affects the environment metadata stored in the Delphix Engine. It will not affect your database installations or homes on the hosts or clusters that the environment is referencing.

Prerequisites

Before you can delete an environment, you must first delete all dependencies such as dSources and virtual databases (VDBs).

Procedure

1. Login to the Delphix Management Application using Delphix Admin credentials.
2. Click **Manage**.
3. Select **Environments**.
4. In the Environments panel, select the environment you want to delete.
5. From the **Actions menu** (**...**) located on the top-right corner select Delete.
6. Click **Yes** to confirm.

Refreshing a DB2 Environment

This topic describes how to refresh an environment.

When a host or cluster has already been configured in the Delphix Management Application as an environment and is subsequently changed in certain ways, an environment refresh may be required for the Delphix Engine to acknowledge the changes. Example actions that could necessitate an environment refresh include:

- Installing a new database home
- Creating a new database
- Adding a new instance

During environment discovery and environment refreshes, the Delphix Engine pushes a fresh copy of the toolkit to each host environment. Included in the toolkit are:

- A Java Runtime Environment, or JRE
- Delphix .JAR files
- The Delphix hostchecker utility
- Scripts for managing the environment and/or VDBs
- Delphix Connector log files (when applicable)

The Delphix Engine then executes some of these scripts to discover information about the objects in your environment, such as where the databases are installed, their names, and information required to connect to them. In some environments (Windows in particular), the scripts are customized to fit the customer’s environment.

Managing DB2 Data Sources

These topics describe special tasks and concepts for linking dSources.

- DB2 dSource Icon Reference
- Linking a dSource from a DB2 Database: An Overview
- Linking a DB2 dSource
- Deleting a DB2 dSource
- Enabling and Disabling DB2 dSources
- Advanced Data Management Settings for DB2 dSources

DB2 dSource Icon Reference

This topic illustrates the icons that appear on dSources and virtual databases (VDBs) in the Delphix Engine graphic user interface and describes the meaning of each, along with tips for clearing those that represent errors.

Icon	Description
⊞	Selecting the Add icon allows you to add a Dataset Group, add a dSource, or Create vFiles
🔍	Search field allows you to search by the name of the dataset, regardless of what group it is in.
▶	Collapses all Groups
▼	Expands all Groups
▶	Collapses the selected group
▼	Expands the selected Group
☰	Icon for CDB - container database
●	Icon for Live Source
☰	Icon for a masked VDB
☰	Icon for a VDB
■	Icon for a vFile
⊞	Icon associated with a warehouse
●	Icon associated with a dSource
📦	Represents a package
🟨	There is a warning fault associated with the dataset
🟥	There is a critical fault associated with the dataset

⚠	There is a critical fault associated with the dSource or VDB. See the error logs for more information.
⚠	There is a warning fault associated with the dSource or VDB. See the error logs for more information.
🛢	The dSource or VDB is ready for Linux Transformation

Linking a dSource from a DB2 Database: An Overview

This topic describes basic concepts behind the creation of dSources from DB2 instances.

Database Level Operation

Associated with an instance is the concept of an instance owner. This is the user who "owns" that instance and has SYSADM authority over the instance and all databases inside that instance. SYSADM authority is the highest level of authority in DB2 and lets this user perform several database management activities such as upgrade, restore, and editing configurations. More information about instances is located in the IBM knowledge center.

Delphix DB2 Instances
Delphix operates at the database level and requires that

- The staging and target hosts must have the empty instances created prior to Delphix using them
- The desired OS user to execute commands related to dSource and VDB operation for each instance has been added as an environment user

Note: Make sure staging instance used for linking doesn't have existing restoring database with the same name.

Data Ingestion

DB2 for Delphix ingests data by using a staging database created on a discovered Standby instance of DB2. The dSource uses the staging database to stay in sync with the production database. This is done by first going through the linking process during which a full backup is used to recover an initial copy of the production database to the staging database. Storage for the staging database is provided by Delphix via an NFS mount to storage exported by the Delphix system.

A single standby instance can contain data from multiple source databases.

Data Synchronization

During the linking process, you can optionally set up an HADR connection between the original source databases and copies on the Standby instance. By doing this the Standby instance will always keep its databases in sync with the source databases using HADR for log shipping. It is important to note that a single Standby instance (dSource) can contain multiple databases from multiple different servers and instances as long as each database has a unique name.

In case of standalone staging host we can use resynchronization utility to keep staging database up to date as source database.

Delphix HADR standby configuration may not meet DR requirements and should not be used as such without evaluating your needs.

Virtual Instance (VDB) Provisioning

A VDB is created based off the desired dSource snapshot and is provisioned to the desired instance on a target system. This operation is executed with the desired target system OS environment user.

A DB2 instance can be used for multiple Delphix objects such as combination of VDBs and dSources.

Related Links

- Requirements for DB2 Hosts and Databases
- DB2 Compatibility Matrix
- Linking a DB2 dSource

Linking a DB2 dSource

This topic describes how to link a DB2 staging dSource.

Prerequisites

- Be sure that the source and staging instances meets the host requirements and the databases meet the container requirements described in Requirements for DB2 Hosts and Databases.

Source Database Preparation

- ***Instance Owner Permissions***

  Delphix uses the DB2 instance owner account on the dSource for many things, including verifying the data inside the databases. For ingesting database on the staging server with different instance we need permissions on the source database to do restore on the staging server. For example in the source if we have an instance named auto1051 and database name delphix and if we want to create a dSource on the auto1052 instance on staging server then you must explicitly grant DBADM and SECADM to the dSource instance auto1052 on the source instance using the following steps:
  1. Connect to the source databases as the source instance owner.
     a. *connect to <DB_NAME> user <INSTANCE_OWNER>*
  2. Issue database grant command
     a. *grant DBADM, SECADM on database to user <DSOURCE_INSTANCE_OWNER>*
  3. Repeat step 2 for every database to be included in the dSource, on the corresponding source database.

Determine if your dSource will be a non-HADR instance, an HADR single standby instance, or an HADR multiple standby instance. Non-HADR dSources can only be updated via a full dSource resync from a newer backup file

Non-HADR Database

1. See "Instance Owner Permissions" section above.
2. Ensure that the source database has the necessary user permissions for the provisioned VDBs as described in Database Permissions for Provisioned DB2 VDBs

HADR Single Standby Database

1. All items in Non-HADR Database section above.
2. The following database configuration settings must be set:
   a. *update db cfg for <DB_NAME> using HADR_LOCAL_HOST <PRIMARY_IP> HADR_LOCAL_SVC <PRIMARY_PORT> immediate*
   b. *update db cfg for <DB_NAME> using HADR_REMOTE_HOST <STANDBY_IP> HADR_REMOTE_SVC <STANDBY_PORT> immediate*
   c. *update db cfg for <DB_NAME> using HADR_REMOTE_INST <STANDBY_INSTANCE_NAME> immediate*
   d. *update db cfg for <DB_NAME> using HADR_SYNCMODE SUPERASYNC immediate*
3. If database configuration parameter LOGINDEXBUILD is set to OFF, do the following:
   i. *update db cfg for <DB_NAME> using LOGINDEXBUILD ON*
   ii. Force off all connections to the database and reactivate the database
4. If database configuration parameter LOGARCHMETH1 is set to OFF, do the following:
   a. *update db cfg for <DB_NAME> using LOGARCHMETH1 XXXX* (must be a valid log archiving method)
   b. Take an offline backup
5. If LOGARCHMETH1 points to a third-party backup server (i.e. TSM or Netbackup) define LOGARCHMETH2 to disk
   a. *update db cfg for <DB_NAME> using LOGARCHMETH2 DISK:<full path to archive log directory>*
      i. Log files in the directory must be available from the time of the backup until the restore has successfully completed on the dSource.
6. *db2 start hadr on db <DB_NAME> as primary by force*
7. Take a full online backup as defined in the "Backup Source Database" section below.
8. Record the following information, as it must be entered on the Delphix Engine while creating the dSource.
   a. HADR Primary hostname
   b. HADR Primary SVC
   c. HADR Standby SVC (auxiliary standby port)

HADR Multiple Standby Databases

This **assumes** a single standby database HADR setup already exists. The existing standby will be referred to as the main standby. The new delphix standby will be referred to as the auxiliary standby.

1. The following database configuration settings must be set on the primary database:
   a. *update db cfg for <DB_NAME> using HADR_SYNCMODE <SYNC MODE> immediate* – set whichever sync mode you wish to use on your main standby.
   b. *update db cfg for <DB_NAME> using HADR_TARGET_LIST "<MAIN_STANDBY_IP:MAIN_STANDBY_PORT|AUXILIARY_STANDBY_IP:AUXILIARY_STANDBY_PORT>"*

immediate

i. You may have up to two auxiliary standbys defined separated by a '|'; one of which must be the delphix dSource.

2. *stop hadr on db <DB_NAME>*
3. *start hadr on db <DB_NAME> as primary by force*
4. Take a full online backup as defined in the "Backup Source Database" section below. While this backup is running, you may continue with step 5.
5. The following database configuration settings must be set on the existing main standby database:
    a. *update db cfg for <DB_NAME> using HADR_SYNCMODE <same mode as defined in 1.a above.>* – It must be the same value used for primary database.
    b. *update db cfg for <DB_NAME> using HADR_TARGET_LIST "<PRIMARY_IP:PRIMARY_PORT|MAIN_STANDBY_IP:MAIN_STANDBY_PORT>"*
6. *stop hadr on db <DB_NAME>*
7. *start hadr on db <DB_NAME> as standby*
8. Record the following information, as it must be entered on the Delphix Engine while creating the dSource (the auxiliary standby database):
    a. HADR Primary hostname
    b. HADR Primary SVC
    c. HADR Standby SVC (auxiliary standby port)
    d. HADR_TARGET_LIST <PRIMARY_IP:PRIMARY_PORT|MAIN_STANDBY_IP:MAIN_STANDBY_PORT>

Backup Source Database

New Feature: Source Database with Raw DEVICE type Storage
Several users use raw device-based tablespaces for source DB2 databases. To leverage these environments with Delphix, Delphix has built a workflow using DB2s native tools that allow Delphix to discover and convert a raw device-based tablespace into an automatic storage-based tablespace during ingestion. Once the data is ingested into staging, customers will be able to provision VDBs of the automatic storage-based database.

In order to complete the linking process, the Standby dSource must have access to a full backup of the source DB2 databases on disk. This should be a compressed online DB2 backup and must be accessible to the dSource instance owner on disk. Delphix is currently not setup to accept DB2 backups taken using third-party sources such as Netbackup or TSM. **Both HADR and Non-HADR backups must also include logs**
.

Example backup command: *db2 backup database <DB_NAME> online compress include logs*

Best Practices for Taking a Backup
The following best practices can help improve backup and restore performance:

1. Compression should be enabled
2. Following parameters should be optimally configured:
    a. Utility Heap Size (UTIL_HEAP_SZ)
    b. No. of CPUs
    c. No. of Table Spaces
    d. Extent Size
    e. Page Size
3. Parallelism & Buffer configuration may be used to improve the backup performance. Parameters that should be configured are :
    a. Parallelism
    b. Buffer Size
    c. No. of Buffers

More information about backup best practices is available in IBM Knowledge Center

Procedure

1. Login to the Delphix Management Application using Delphix Admin credentials or as the owner of the database from which you want to provision the dSource.
2. On the **Databases** tab of Environment Management screen, add a source config against discovered staging instance.
3. Then, click **Manage**.
4. Select **Datasets**.
5. Click the Plus (+) icon and select **Add dSource**, you'll get a list of available source configs using which you can go for dsource creation.
6. In the **Add dSource** wizard, select the required source configuration.
7. If you are working with an HADR setup, please leave the HADR checkbox checked.
8. The **database name** is mandatory and must be unique for a given instance. This is the name that the database was on the instance it

was restored from.

9. Enter the complete **Backup Path** where the database backup file resides. If no value is entered, the default value used is the instance home directory. If there are multiple backup files for a database on the backup path, the most current one will be used.
10. Enter the **Log Archive Method1** you wish to use for the database. If no value is entered, the default value used is DISK:/mountpoint/dbname/arch.
11. Optionally, users can set the database configuration parameters during the linking operation in the **Config Settings** section.
12. If the dSource is to use HADR please enter the following fields. If it will not use HADR skip ahead to step 13. For more information about HADR please view Linking a dSource from a DB2 Database: An Overview.

    a. Enter a fully qualified HADR Primary Hostname. This is a required field for HADR and must match the value set for HADR_LOCAL_HOST on the master.

    b. Enter the port or /etc/services name for the HADR Primary SVC. This is a required field for HADR and uses the value set for HADR_LOCAL_SVC on the master.

    c. Enter the port or /etc/services name for the HADR Standby SVC. This is a required field for HADR and uses the value set for HADR_REMOTE_SVC on the master.
13. Click **Next**.
14. Select a **dSource Name** and **Database Group** for the dSource.
15. Click **Next**.

    You will get Data Management section where you need to specify staging environment and user which will be used for dsource creation.
16. Set the **Staging Environment** to be the same as the dSource host.
17. Select the **Staging Environment User** to be the same as the instance owner of the dSource instance.

    Changing the Environment User
    If you need to change or add an environment user for the dSource instance, see Managing DB2 Users and Instance Owners.

18. Then, click **Next** and you'll get Policies section. Set the desired **Snapsync Policy** for the dSource. For more information on policies see Advanced Data Management Settings for DB2 dSources.
19. Click **Next**.
20. Specify any desired pre- and post-scripts. For details on pre- and post-scripts, refer to Customizing DB2 Management with Hook Operations.
21. Click **Next**.
22. Review the dSource Configuration and Data Management information in summary section.
23. Click **Submit**.

The Delphix Engine will initiate two jobs to create the dSource: DB_Link and DB_Sync. You can monitor these jobs by clicking Active Jobs in the top menu bar, or by selecting System > Event Viewer. When the jobs have completed successfully, the database icon will change to a dSource icon on the Environments > Host > Databases screen, and the dSource will also appear in the list of Datasets under its assigned group.

The dSource Configuration Screen
After you have created a dSource, the **dSource Configuration tab** allows you to view information about it and make modifications to its policies and permissions. In the **Datasets** panel, select the dSource you wish to examine. You can now choose the configuration tab to see information such as the **Source files, Data Management** configuration and **Hook Operations**. For more information, see Advanced Data Management Settings for DB2 dSources.

Deleting a DB2 dSource

Deleting a dSource will delete the dSource metadata for a particular source database, along with all snapshots, logs, and policies stored in Delphix. This is a permanent operation, and re-attaching the dSource will require a new linking operation. If a dSource is deleted, it does not affect your source database.

If you wish to temporarily disable your dSource without deleting it, you can follow the steps to Enabling and Disabling dSources instead.

Prerequisites

You cannot delete a dSource that has dependent virtual databases (VDBs). Before deleting a dSource, make sure that you have deleted all dependent VDBs as described in Deleting a VDB.

Procedure

1. Login to the **Delphix Management** application using Delphix Admin credentials.
2. Click **Manage**.
3. Click **Datasets**.
4. In the **Datasets** list, select the **dSource** you want to delete.

5. From the **Actions** menu (...) select **Delete.**
6. Click **Delete** to confirm.

Deleting a dSource will also delete all snapshots, logs, and descendant VDB refresh policies for that dSource. You cannot undo the deletion.

Enabling and Disabling DB2 dSources

This topic describes how to enable and disable dSources when certain operations against the source database must occur outside of Delphix.

Some operations, such as restoring the source database from a backup, will require that the dSource be temporarily disabled. Disabling a dSource turns off communication between it and the source database, but it does not tear down the configuration that enables communication and data updating to take place. When a disabled dSource is later enabled, it will resume communication and incremental data updates from the source database according to the original policies and data management configurations that you set.

Disabling a dSource is also a prerequisite for several other operations, such as database migration and upgrading the dSource metadata after upgrade of the associated data source.

Procedure

Disabling a dSource will stop further operations on the Delphix Engine related to the dSource.

1. Login to the Delphix Management application as **delphix_admin** or another user with administrative privileges.
2. Click **Manage**.
3. Select **Datasets**.
4. Select the **dSource** you want to disable.
5. In the upper right-hand corner, from the **Actions** menu (...) select **Disable.**
6. In the Disable dialog select **Disable**.

When you are ready to enable the dSource again, from the Actions menu (...) select **Enable**, and the dSource will continue to function as it did previously.

Optionally, you can edit the database configuration parameters which were provided during the linking operation.

Advanced Data Management Settings for DB2 dSources

- Accessing Data Management Settings
- Retention Policies
- Benefits of Longer Retention

This topic describes advanced data management settings for dSources.

When linking a dSource, you can use custom data management settings to improve overall performance and match the needs of your specific server and data environment. If no specific settings are required, leverage default data management settings.

Accessing Data Management Settings

There are three ways to set or modify data management settings for dSources:
1. During the dSource linking process, in the Policies tab of the Add dSource wizard select the **Retention Policy**.
2. Or select **Manage**, then select **Policies**.
3. In the Policies screen select the **Retention** tab.
4. To add a new retention policy click the **+Retention** button.
5. This will open the Retention Policy wizard. Enter a policy name, select your retention periods and click **Submit**.
   For more information, see Creating Custom Policies .

Retention Policies

Retention policies define the length of time Delphix Engine retains snapshots within its storage. To support longer retention times, you may need to allocate more storage to the Delphix Engine. The retention policy – in combination with the SnapSync policy – can have a significant impact on the performance and storage consumption of the Delphix Engine.

Benefits of Longer Retention

With increased retention time for snapshots and logs, you allow a longer (older) rollback period for your data.

Common use cases for longer retention include:

- SOX compliance
- Frequent application changes and development
- Caution and controlled progression of data
- Reduction of project risk
- Restore to older snapshots

Provisioning VDBs from DB2 dSources

These topics describe special tasks and concepts for provisioning VDBs from dSources.

- Provisioning DB2 VDBs: An Overview
- Provisioning a DB2 VDB
- Database Permissions for Provisioned DB2 VDBs
- Upgrading DB2 VDBs
- Rewinding a DB2 VDB

Provisioning DB2 VDBs: An Overview

This topic describes the basic concepts involved with provisioning VDBs from DB2.

A dSource is a virtualized representation of a physical or logical source database. As a virtual representation, it cannot be accessed or manipulated using database tools. Instead, you must create a virtual database (VDB) from a dSource snapshot. A VDB is an independent, writable copy of a dSource snapshot. You can also create VDBs from other VDBs. Once you have provisioned a VDB to a target environment, you can also implement snapshot and retention policies for the VDB, which will determine how frequently Delphix Engine will take a database snapshot and how long the snapshots will be retained for recovery and provisioning purposes.

For an overview of the high-level components involved in provisioning a DB2 VDB refer to Setting Up DB2 Environments: An Overview.

Snapshots accumulate over time. To view a snapshot:

1. From the Datasets panel, click the group containing the dSource.
2. Select the dSource.
3. Click the TimeFlow tab.

The TimeFlow appears as a list of dates, each of which expands to show snapshots from that date. Times when the VDB has been refreshed are marked by a blue line between dates.

On the TimeFlow, you can also filter by type of snapshot. To do so, click the filter button, which is shaped like an eye.

You can scroll through these lists to select the one you want, or you can enter a date and time to search for a specific snapshot.

Once you have provisioned a VDB, you can also take snapshots of it. As with the dSource snapshots, you can find these when you select the VDB in the Datasets panel. You can then provision additional VDBs from these VDB snapshots.

Dependencies
If there are dependencies on the snapshot, you will not be able to delete the snapshot free space; the dependencies rely on the data associated with the snapshot.

Related Links

- Setting Up DB2 Environments: An Overview
- Provisioning a DB2 VDB
- Database Permissions for Provisioned DB2 VDBs

Provisioning a DB2 VDB

This topic describes how to provision a virtual database (VDB) from a DB2 dSource.

Prerequisites

- You will need to have linked a dSource from a staging instance, as described in Linking a DB2 dSource, or have created a VDB from which you want to provision another VDB

- You should have set up the DB2 target environment with necessary requirements as described in Requirements for DB2 Hosts and Databases
- Make sure you have the required Instance Owner permissions on the target instance and environment as described in Managing DB2 Users and Instance Owners
- The method for Database Permissions for Provisioned DB2 VDBs is decided before the provisioning

You can take a new snapshot of the dSource by clicking the **Camera** icon on the dSource card. Once the snapshot is complete you can provision a new VDB from it.

Procedure

1. Login to the **Delphix Admin** application.
2. Click **Manage**.
3. Select **Datasets**.
4. Select a **dSource**.
5. Select a **snapshot** from which you want to provision.
6. Click **Provision VDB** icon to open Provision VDB wizard.
7. Select a target environment from the left pane.
8. Select an **Installation** to use from the dropdown list of available DB2 instances on that environment.
9. Set the **Environment User** to be the Instance Owner. Note: The picking of instance owner is only possible if you have multiple environment users set on that host.
10. Provide VDB Name as database name as parameter.
11. Optionally, set the database configuration parameters for the VDB.
12. Click **Next**.
13. Select a **Target Group** for the VDB.
    Click the green **Plus** icon to add a new group, if necessary.
14. Select a **Snapshot Policy** for the VDB.
15. Click **Next**.
16. Specify any desired hook operations. For details on hook operations, refer to Customizing DB2 Management with Hook Operations.
17. Click **Next**.
18. Review the **Provisioning Configuration** and **Data Management** information.
19. Click **Submit**.
    When provisioning starts, you can review the progress of the job in the **Databases** panel, or in the **Job History** panel of the **Dashboard**. When provisioning is complete, the VDB will be included in the group you designated, and listed in the **Databases** panel. If you select the VDB in the Databases panel and click the **Open** icon, you can view its card, which contains information about the database and its Data Management settings.
    Once the VDB provisioning has successfully completed, if the source and target instance ids are not the same, you may want to grant secadm and dbadm on the database to the target instance id. Refer to Database Permissions for Provisioned DB2 VDBs for more information.

Related Links

- Linking a DB2 dSource
- Provisioning DB2 VDBs: An Overview
- Database Permissions for Provisioned DB2 VDBs
- Customizing DB2 Management with Hook Operations

Database Permissions for Provisioned DB2 VDBs

This topic describes the database permissions on provisioned DB2 virtual instances

DB2 Authentication

Authentication is the process of validating a supplied user ID and password using a security mechanism. User and group authentication is managed in a facility external to DB2 LUW, such as the operating system, a domain controller, or a Kerberos security system. This is different from other database management systems (DBMSs), such as Oracle and SQL Server, where user accounts may be defined and authenticated in the database itself, as well as in an external facility such as the operating system.

Any time a user ID and password is explicitly provided to DB2 LUW as part of an instance attachment or database connection request, DB2 attempts to authenticate that user ID and password using this external security facility. If no user ID or password is provided with the request, DB2 implicitly uses the user ID and password that were used to login to the workstation where the request originated. More information on DB2 authentication and authorization is available via IBM documentation.

Delphix DB2 Authentication
Delphix for DB2 requires that that the staging and target hosts must already have the necessary users and authentication systems created/installed on them. Delphix will neither create users nor change database passwords as part of the provisioning process.

While the terminology used within the Delphix Management application refers to a VDB, the ingestion, snapshot and provisioning process for DB2 on Delphix always occurs on the Database level. Thus when a virtual DB2 database is provisioned by Delphix, it contains the DB2 database that were in the source instance with the identical user permissions as they had on the source. This means that if the target instance name is different from the source instance name then that instance owner will NOT have DBADM or SECADM permissions unless they were specifically granted to that instance owner on the source instance. The instance owner will however always have SYSADM permissions on all databases in the instance.

LDAP Authentication

If your DB2 instances and applications use LDAP authentication then they will work seamlessly as long as LDAP had been configured on the VDB Target Instance.

OS Authentication

If your DB2 instances and applications are using OS authentication then it is important to ensure that the relevant OS accounts exist on the target machine.

Generic Accounts

If the DB2 applications are using generic (non-instance owner) accounts, they will then be able to continue using them as long as those OS accounts exist on the host machine. It is important to note that the passwords for the same account may be different on different hosts, or if they use different LDAP servers (i.e. prod vs dev ldap servers).

Instance Owner Usage

If the DB2 applications typically use the instance owner accounts to access data then it is important to note that if the new instance name is different from the source instance name then that instance owner will NOT have DBADM or SECADM permissions in the VDB instance and may not be able to read and write data. In such a case there are three possible workarounds:

1. Connect to the databases as the source instance owner. This requires that the target host have a unix account with the same name as the source instance.
2. Grant the necessary permissions to the VDB instance owner on the source instance before taking the snapshot that is to be provisioned.
3. Grant the permissions to a known "delphix" user on the source instance and then use that account to grant the permissions to the target VDB instance after provisioning.

Related Links

- Managing DB2 Instances
- Managing DB2 Users and Instance Owners
- Provisioning VDBs from DB2 dSources

Upgrading DB2 VDBs

This topic describes how to upgrade a DB2 VDB to a higher version of DB2.

Procedure for VDB In-Place Upgrade

1. Remove any VDB Refresh Policy assigned to the VDB.
2. Upgrade the target DB2 instance.
3. Refresh the target environment.

Rewinding a DB2 VDB

This topic describes the procedure for rewinding a VDB.

Rewinding a VDB rolls it back to a previous point in its TimeFlow and re-provisions the VDB. The VDB will no longer contain changes after the rewind point.

> Although the VDB no longer contains changes after the rewind point, the rolled over Snapshots and TimeFlow still remain in Delphix and are accessible through the Command Line Interface (CLI). See the topic CLI Cookbook: Rolling Forward a VDB for instructions on how to use these snapshots to refresh a VDB to one of its later states after it has been rewound.

> Delphix clones a new Timeflow from the closest Snapshot older than or equal to the rewind point. This creates a dependency between the new Timeflow and the parent Snapshot and Timeflow. The parent Snapshot and Timeflow cannot be deleted because of this dependency. The VDB must first be refreshed before the parent Snapshot and Timeflow can be removed.

Prerequisites

To rewind a VDB, you must have the following permissions:

- **Auditor** permissions on the dSource associated with the VDB
- **Owner** permissions on the VDB itself

You do NOT need owner permissions for the group that contains the VDB. A user with Delphix Admin credentials can perform a VDB rewind on any VDB in the system.

Procedure

1. Login to the **Delphix Admin** application.
2. Under **Datasets**, select the **VDB** you want to rewind.
3. On the **Timeflow** tab, select the rewind point as a snapshot or a point in time.
4. Click **Rewind**.
5. If you want to use login credentials on the target environment other than those associated with the environment user, click **Provide Privileged Credentials**.
6. Click **Yes** to confirm.

> **Using TimeFlow Bookmarks**
> You can use a TimeFlow bookmark as the rewind point when using the CLI. Bookmarks can be useful to:
>
> - Mark where to rewind to – for example, before starting a batch job on a VDB
> - Provide a semantic point to revert back to in case the chosen rewind point turns out to be incorrect.
>
> For a CLI example using a TimeFlow bookmark, see CLI Cookbook: Provisioning a VDB from a TimeFlow Bookmark.

Customizing DB2 Management with Hook Operations

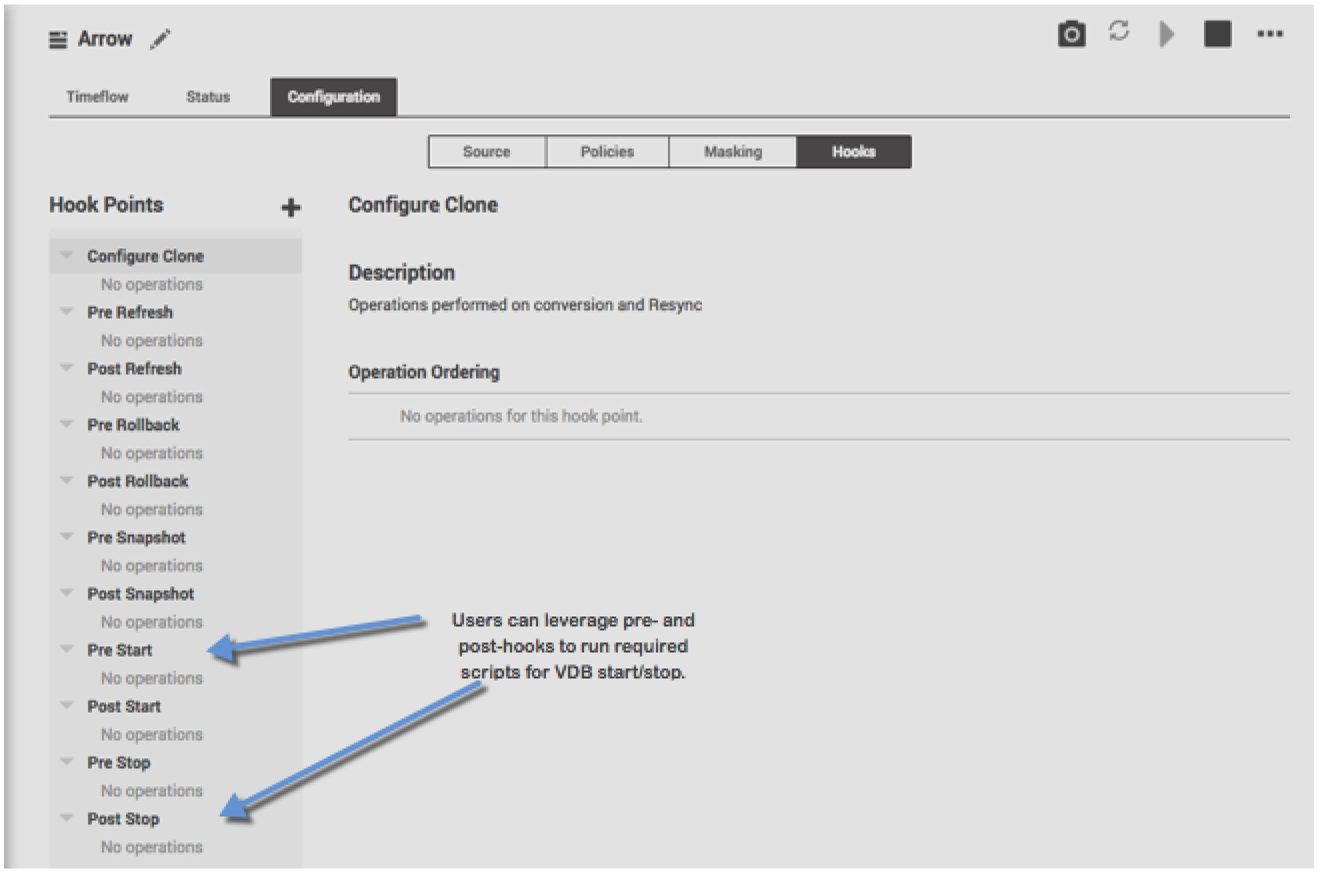
Hook operations allow you to execute an ordered list of custom operations at select hook points in linking, provisioning and virtual dataset management. For details on the types of operations that are available, see children of this page.

VDB and vFile Hooks

Hook	Description
Configure Clone	Operations performed after initial provision or after a refresh. This hook will run after the virtual dataset has been started. During a refresh, this hook will run before the Post-Refresh hook.
Pre-Refresh	Operations performed before a refresh. This hook will run before the virtual dataset has been stopped. These operations can cache data from the virtual dataset to be restored after the refresh completes.

Post-Refresh	Operations performed after a refresh. This hook will run after the virtual dataset has been started and after the Configure Clone hook. This hook will not run if the refresh or Pre-Refresh hook operations fail. These operations can restore cached data after the refresh completes.
Pre-Rewind	Operations performed before a rewind. This hook will run before the virtual dataset has been stopped. These operations can cache data from the virtual dataset to be restored after the rewind completes.
Post-Rewind	Operations performed after a rewind. This hook will not run if the rewind or Pre-Rewind hook operations fail. This hook will run after the virtual dataset has been started. This hook will not run if the rewind or Pre-Rewind hook operations fail. These operations can restore cached data after the rewind completes.
Pre-Snapshot	Operations performed before a snapshot. These operations can quiesce data to be captured during the snapshot, or stop processes that may interfere with the snapshot.
Post-Snapshot	Operations performed after a snapshot. This hook will run regardless of the success of the snapshot or Pre-Snapshot hook operations. These operations can undo any changes made by the Pre-Snapshot hook.
Pre-Start	Operations performed before startup of a VDB or vFile. These operations can be used to initialize configuration files, or stop processes that might interfere with the virtual dataset.
Post-Start	Operations performed after startup of a VDB of vFile. These operations can be used to clean up any temporary files, or restart processes that may have been stopped by a Pre-Start hook, or log notifications.
Pre-Stop	Operations performed before shutdown of a VDB or vFile. These operations can quiesce data or processes prior to virtual dataset shutdown.
Post-Stop	Operations performed after shutdown of a VDB or vFile. These operations can be used to log notifications, clean up any temporary files, or stop/restart related processes.

You can leverage hooks to run required scripts which address several different use cases. For example, you may want to prevent your monitoring systems from triggering during VDB startup and shutdown. As shown in the figure below, you can now leverage pre- and post-hooks to run required scripts for VDB start/stop operations.



Operations performed on conversion and Resync

No operations for this hook point.

Users can leverage pre- and post-hooks to run required scripts for VDB start/stop.

Hooks

Operation Failure
If a hook operation fails, it will fail the entire hook: no further operations within the failed hook will be run.

Setting Hook Operations

You can construct hook operation lists through the Delphix Management application or the command line interface (CLI). You can either define the operation lists as part of the provisioning process or edit them on virtual datasets that already exist.

Setting Hook Operations through the Delphix Management Application

The provisioning wizard still imports hook operations from templates.

Hook operations can be provisioned in the **Hooks** tab of the Add dSource or Add VDB wizards.

1. Select the **type of operation** and enter a name, operation type, and script.
2. To remove an operation from the list, click the **Trash** icon on the operation.
3. When you have set all hook operations, click **Next** to continue with the provisioning process.

To edit hook operations on a virtual dataset:

From the **Datasets** panel, you can create hook operations from a template.

1. In the **Datasets** panel, click the virtual dataset.
2. Click the **Configuration** tab.
3. Within the **Configuration** tab, click the **Hooks** tab.
4. Select the **hook** to edit.

5. The current operations at this hook will be displayed. To edit this list of operations, click the **Pencil** icon in the top right-hand corner of the tab.
6. Click the **Plus** icon to add a new operation.
7. Select the **type of operation** or click
   to load a hook operation template.
8. Click the **text area** and edit the contents of the operation.
9. To remove an operation from the list, click the **Trash** icon on the operation.
10. When you have set all hook operations, click the **checkmark** to save the changes.

Setting Hook Operations through the CLI

To specify hook operations during linking, edit the relevant hook's array of operations defined on the `LinkingParameters > Source > Operations` object.

To specify hook operations during provisioning, edit the relevant hook's array of operations defined on the `ProvisionParameters > Source > Operations` object.

To edit hook operations on a dSource that already exists, edit the relevant hook's array of operations defined on the `Source > Operations` object.

To edit hook operations on a virtual dataset that already exists, edit the relevant hook's array of operations defined on the `Source > Operations` object.

For more information about these CLI objects, see the following documentation in the **Help** menu of the **Delphix Management** application:

- `LinkedSourceOperations`
- `VirtualSourceOperations`
- `RunCommandOnSourceOperation`
- `RunExpectOnSourceOperation` API

Example of Editing Hook Operations through the CLI

1. Navigate to the relevant source's **`VirtualSourceOperations`** object.
2. Select a **hook** to edit.

```
delphix> source
delphix source> select "pomme"
delphix source "pomme"> update
delphix source "pomme" update *> edit operations
delphix source "pomme" update operations *> edit postRefresh
```

3. Add an operation at index 0.

```
delphix source "pomme" update operations postRefresh *> add
delphix source "pomme" update operations postRefresh 0 *> set
type=RunCommandOnSourceOperation
delphix source "pomme" update operations postRefresh 0 *> set
command="echo Refresh completed."
delphix source "pomme" update operations postRefresh 0 *> ls
Properties
    type: RunCommandOnSourceOperation (*)
    command: echo Refresh completed. (*)
delphix source "pomme" update operations postRefresh 0 *> commit
```

4. Add another operation at index 1 and then delete it.

```
delphix source "pomme" update operations postRefresh *> add
delphix source "pomme" update operations postRefresh 1 *> set
type=RunCommandOnSourceOperation
delphix source "pomme" update operations postRefresh 1 *> set
command="echo Refresh completed."
delphix source "pomme" update operations postRefresh 1 *> back
delphix source "pomme" update operations postRefresh *> unset 1
delphix source "pomme" update operations postRefresh *> commit
```

Hook Operation Templates

You can use templates to store commonly used operations, which allows you to avoid repeated work when an operation is applicable to more than a single virtual dataset. You manage templates through the Delphix Management application.

Hook Operations Templates Not Available via CLI
Hook operation templates cannot be fully utilized from the CLI. Manage and use hook operations through the Delphix Management application.

Creating a Hook Operation Template

The provisioning wizard still imports hook operations from templates.

1. Login to the **Delphix Management** application using Delphix Admin credentials.
2. Click **Manage**.
3. Select **Operation Templates**.
4. Click the **Plus** icon to add a new operation template.
5. Enter a **Name** for the template.
6. Select an operation **Type**.
7. Enter a **Description** detailing what the operation does or how to use it.
8. Enter operation **Contents** to implement the operation partially or fully.
9. Click **Create**.

Importing a Hook Operation Template

To import a hook operation template:

1. In the **Datasets** panel, select a dataset.
2. Click the **Configuration** tab.
3. Within the **Configuration** tab, click the **Hooks** tab.
4. Select the **hook** to edit.
5. Click the **Plus** icon to add a new operation.
6. Click **Import**.
7. Select the **template** to import.
8. Click **Import**.
9. When you have set all hook operations, click **Check** to save the changes.

Exporting a Hook Operation Template

To export a hook operation template:

1. In the **Datasets** panel, select a dataset.
2. Click the **Configuration** tab.
3. Within the **Configuration** tab, click the **Hooks** tab.
4. Select the **hook** to edit.
5. Click the **Plus** icon to add a new operation.

6. Select the **type of operation**.
7. Click the **text area** and edit the contents of the operation.
8. Click **Export**.
9. Enter a **Name** for the template.
10. Enter a **Description** detailing what the operation does or how to use it.
11. Click **Export.**

DB2 Hook Operation Notes

Shell Operations

RunCommand Operation

The RunCommand operation runs a shell command on a Unix environment using whatever binary is available at `/bin/sh`. The environment user runs this shell command from their home directory. The Delphix Engine captures and logs all output from this command. If the script fails, the output is displayed in the Delphix Management application and command line interface (CLI) to aid in debugging.

If successful, the shell command must exit with an exit code of `0`. All other exit codes will be treated as an operation failure.

Examples of RunCommand Operations

You can input the full command contents into the RunCommand operation.

```
remove_dir="$DIRECTORY_TO_REMOVE_ENVIRONMENT_VARIABLE"

if test -d "$remove_dir"; then
    rm -rf "$remove_dir" || exit 1
fi

exit 0
```

If a script already exists on the remote environment and is executable by the environment user, the RunCommand operation can execute this script directly.

```
/opt/app/oracle/product/10.2.0.5/db_1/dbs/myscript.sh
"$ARG_ENVIRONMENT_VARIABLE" "second argument in double quotes" 'third
argument in single quotes'
```

RunBash Operation

The RunBash operation runs a Bash command on a Unix environment using a `bash` binary provided by the Delphix Engine.The environment user runs this Bash command from their home directory. The Delphix Engine captures and logs all output from this command. If the script fails, the output is displayed in the Delphix Management application and command line interface (CLI) to aid in debugging.

If successful, the Bash command must exit with an exit code of `0`. All other exit codes will be treated as an operation failure.

Example of RunBash Operations

You can input the full command contents into the RunBash operation.

```
remove_dir="$DIRECTORY_TO_REMOVE_ENVIRONMENT_VARIABLE"

# Bashisms are safe here!
if [[ -d "$remove_dir" ]]; then
    rm -rf "$remove_dir" || exit 1
fi

exit 0
```

Shell Operation Tips

Using nohup

You can use the `nohup` command and process backgrounding from resource in order to "detach" a process from the Delphix Engine. However, if you use `nohup` and process backgrounding, you MUST redirect `stdout` and `stderr`.

Unless you explicitly tell the shell to redirect `stdout` and `stderr` in your command or script, the Delphix Engine will keep its connection to the remote environment open while the process is writing to either `stdout` or `stderr` . Redirection ensures that the Delphix Engine will see no more output and thus not block waiting for the process to finish.

For example, imagine having your `RunCommand` operation background a long-running Python process. Below are the bad and good ways to do this.

Bad Examples

- `nohup python file.py & # no redirection`
- `nohup python file.py 2>&1 & # stdout is not redirected`
- `nohup python file.py 1>/dev/null & # stderr is not redirected`
- `nohup python file.py 2>/dev/null & # stdout is not redirected`

Good Examples

- `nohup python file.py 1>/dev/null 2>&1 & # both stdout and stderr redirected, Delphix Engine will not block`

Other Operations

RunExpect Operation

The RunExpect operation executes an Expect script on a Unix environment. The Expect utility provides a scripting language that makes it easy to automate interactions with programs which normally can only be used interactively, such as `ssh`. The Delphix Engine includes a platform-independent implementation of a subset of the full Expect functionality.

The script is run on the remote environment as the environment user from their home directory. The Delphix Engine captures and logs all output of the script. If the operation fails, the output is displayed in the Delphix Management application and CLI to aid in debugging.

If successful, the script must exit with an exit code of `0`. All other exit codes will be treated as an operation failure.

Example of a RunExpect Operation

Start an `ssh` session while interactively providing the user's password.

```
spawn ssh user@delphix.com
expect {
    -re {Password: } {
        send "${env(PASSWORD_ENVIRONMENT_VARIABLE)}\n"
    }
    timeout {
        puts "Timed out waiting for password prompt."
        exit 1
    }
}
exit 0
```

DB2 Environment Variables

Operations that run user-provided scripts have access to environment variables. For operations associated with specific dSources or virtual databases (VDBs), the Delphix Engine will always set environment variables so that the user-provided operations can use them to access the dSource or VDB.

dSource Environment Variables

Environment Variable	Description
`DLPX_DATA_DIRECTORY`	Path where staging database is mounted

VDB Environment Variables

Environment Variable	Description
`DLPX_DATA_DIRECTORY`	Path where virtual database is mounted