HOOKS IN DELPHIX

Version: 1.0

Abstract

This document details the hooks concept, types of hooks, operations available, and their usage in the Delphix Engine

For more information on hooks, see the following pages in Delphix documentation:
Customizing Oracle Management with Hook Operations
Oracle Hook Operation Notes

Contents

Hooks

A hook is a special kind of code snippet that automatically executes in response to a certain event that occurs in the Delphix Engine. It can be in-place code, or it can call an existing script on the target server. Conceptually, a hook is comparable to triggers in RDBMS databases at database level, but with more flexibility.

Hooks allow an ordered list of user-defined operations to execute sequentially as part of the virtual dataset provisioning and refresh processes. You can access hooks in the **Hooks** tab of the provisioning wizard, and from the **Configuration** tab of a selected virtual dataset.

Hooks are mainly used for pre- and post-provisioning operations. For example, you can use hooks to:

- back up test data before refresh and restore
- back up data after provisioning
- create logins for dev/qa users who do not have privileges on production databases
- reset passwords of users cloned from production database
- back up configuration data from database and many more use cases.

In the Delphix Engine, there are two different types of dSources hooks, seven different types of virtual dataset hooks, and two different types of hook operations.

Types of dSource Hooks

This type of hook executes during the Snapsync operation of a dSource.

Pre-Sync

Pre-sync hook operations are executed before a Snapsync of a dSource. These operations can quiesce data to be captured during the Snapsync, or stop application processes that may interfere with the Snapsync operation.

Post-Sync

Post-Sync hook operations are executed after a Snapsync operation of a dSource. These operations can undo any changes made by the pre-sync hooks. They will run regardless of the success of pre-sync hook operations or the Snapsync itself.

Types of Virtual Dataset Hooks

Configure Clone

Configure clone hook operations are executed during initial creation of a virtual dataset and after every refresh. During any refresh, the Delphix Engine will always execute this hook before the post-refresh hook (see below).

Pre-Refresh

Pre-refresh hook operations are executed before a refresh of a virtual dataset. These operations can back up any data or configuration from the running dataset before refreshing.

Post-Refresh

Post-refresh hook operations are executed after a refresh of a virtual dataset. You can use these operations to restore any data or configuration backed up by the pre-refresh hooks. During a refresh, the Delphix Engine runs this hook after the Configure Clone hook.

Pre-Rewind

Pre-rewind hook operations are executed before a rewind of a virtual dataset. These operations can back up any data or configuration from the running dataset before rewinding.

Post-Rewind

Post-rewind hook operations are executed after a rewind of a virtual dataset. You can use these operations to rewind any data or configuration backed up by the pre-rewind hooks. During a rewind, the Delphix Engine runs this hook after the configure clone hook.

Pre-Snapshot

Pre-snapshot hook operations are executed before taking a snapshot of a virtual dataset. These operations can quiesce data to be captured during the snapshot, or stop processes that may interfere with the snapshot.

Post-Snapshot

Post-snapshot hook operations are executed after taking a snapshot of a virtual dataset. These operations can undo any changes made by the pre-snapshot hooks. This hook will run regardless of the success of the pre-snapshot hook operation or the snapshot itself.

Types of Hook Operations

Each hook operation represents a user-configurable action that the Delphix Engine will take. You can configure the operations to fail if they encounter an unexpected error. The failure of a hook operation will cause the enclosing provision or refresh job to halt.

System Shell Command (RunCommand Operation)

The System Shell Command operation runs a shell script on the target host using the environment user's default system shell. The environment user runs the shell command from their home directory. The Delphix Engine captures and logs all output of the script.

If a series of System Shell Command operations is rather long, it might be worth putting the equivalent logic in a script accessible on the target environment. This script can be run by a single System Shell Command operation.

The shell command must exit with an exit code of 0 if successful. All other exit codes will be treated as an operation failure.

Bash Shell Command (RunBash Operation)

This is exact same as System Shell Command with one difference: the Delphix Engine will execute commands using bash shell. Bash shell should be available on target host. If the default shell of environment user is bash, then System Shell Command and Bash Shell Command are equivalent.

Expect Script (RunExpect Operation)

The Expect Script operation executes an Expect script on the target host. The Expect utility provides a scripting language that makes it easy to automate interactions with programs which you can normally only use 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 target environment as the environment user from their home directory. The Delphix Engine captures and logs all output of the script.

The script must exit with an exit code of 0 if successful. All other exit codes will be treated as an operation failure.

PowerShell Script (RunCommand Operation)

The PowerShell Command operation is designed to support Windows environments. The environment user runs the PowerShell command, which runs a PowerShell script on the target host. The Delphix Engine captures and logs all output of the script.

If a series of System Shell Command operations is rather long, it might be worth putting the equivalent logic in a PowerShell script accessible on the target environment. This script can be run by a single PowerShell Command operation.

The PowerShell command must exit with an exit code of 0 if successful. All other exit codes will be treated as an operation failure.

Notes

- If a hook operation fails, it will fail the entire sync, provision, rewind, refresh, or snapshot job. The job will halt immediately and no further hook operations will be run.
- If you want the provision or refresh processes to succeed regardless of the success of the hook operations, ensure that every operation can only exit with an exit code of 0.
- When linking from, or provisioning to, cluster environments such as Oracle RAC environments, hook operations will not run once on each node in the cluster. Instead, the Delphix Engine picks a node in the cluster at random and guarantees that all operation within any single hook will execute serially on this node. Note that the Delphix Engine does not guarantee that the same node is chosen for the execution of every hook.

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 dSource or virtual dataset. You can manage templates through the Delphix Admin application.

You can also create templates from existing hooks by exporting the hooks in the Delphix Admin application.

Environment Variables and Sample Hooks

Delphix-provided environment variables are available inside each Hook operation. See the database-specific Hook documentation for more information on what these variables contain.
Please refer delphix docs for sample hooks and usage of environment variables.