hp StorageWorks multi-site disaster tolerant solution

implementation blueprint

multi-site DT management tools user guide

notice

Edition 0403

format conventions

note This is a Note.

caution This is a Caution.

`User Input`	Specifies text to be typed exactly as shown, such as commands, path names, file names, and directory names.
`variable`	Indicates that you must supply a value.
`Screen text`	Denotes text displayed on the screen.

contents

1 Introduction

The XP Multi-Site DT Management Tools are part of an HP Multi-Site Disaster Tolerant Solution (MSDT). The Tools supply a command line interface (CLI) that allows you to send commands to all of your data sites from a remote location.

The Tools consist of a CLI, meta-commands, and sample scripts that enable you to:

- Use meta-commands to execute RAID Manager (RM) commands on remote hosts, using host and instance information specified in the Tools configuration file. Meta-commands automatically execute on alternative servers when the primary server is unavailable.
- Establish secure connections between two or more hosts.
- Use predefined scripts to combine multiple commands that perform operations for the entire solution.
- Perform script operations from a single host, allowing you to manage a complex environment from a single system.

purpose

This user guide explains how to install and configure the Tools, and describes the configuration file, commands, and scripts that the Tools use to communicate with the various hosts in the solution.

audience

This user guide is intended for:

- HP Account Service Software engineers (ASEs)
- HP Consulting and Integration engineers
- Customer administrators of the MSDT Solution

required skills and knowledge

Users of the MSDT Tools must have proficiency with:

- HP StorageWorks RAID Manager XP
- HP StorageWorks Continuous Access and Continuous Access Extension
- HP StorageWorks Business Copy
- The MSDT Solution configuration
- Basic Unix scripting

2 Installing and Configuring the MSDT Tools

To begin using the Tools, you must install the appropriate software on the designated HP-UX systems, and make any configuration adjustments necessary to establish communication through firewalls in your network.

software requirements

The Tools CLI Server is a plug-in component for the HP OpenView Storage Area Manager (SAM) framework (HP -SAM HostAgent). The -SAM HostAgent is already installed if the system includes any of the following products:

- HP OpenView StorageArea Manager version 3.0
- HP StorageWorks Command View SDM version 1.06
- HP Command View XP version 1.60

The Tools require -SAM Host Agent version 3.0. The Tools installation script (see below) automatically checks for the existence of the SAM Host Agent v3.0 on the system and executes the appropriate action:

- If the SAM HostAgent v3.0 is not installed, the installation script installs the correct SAM Host Agent in addition to the Tools.
- If the SAM Host Agent v3.0 is already installed on the system, the installation script installs only the Tools.
- If an earlier version of the SAM HostAgent is installed on the system, the installation script returns an error message and exits without installing the Tools. In this case you must upgrade the SAM HostAgent to v3.0 before installing the Tools.

installing the tools

You can install the Tools from an installation disk or by downloading them from the HP Storage Tools website.

option 1: install from a CD-ROM script:

1. On a HP-UX system designated as your local system, log in as **`root`**.
2. Create the mount point:

   `/cdrom (# mkdir /cdrom)`
3. Insert the MSDT Tools CD-ROM into a drive connected to your system. You must mount the CD manually. Example:

   `# mount –o ro /dev/dsk/c0t6d0 /cdrom` where **`/dev/dsk/c0t6d0`** is the device file name for the CD-ROM drive.
4. Type the following command to start the installation script:

   `# cd /cdrom`

   `# ./install_hpux_msdt`

   The installation script launches, checks for the existence of SAM Host Agent version 3.0 (see above), and prompts you for information.
5. Enter, individually, the IP addresses of each host running the MSDT CLI server that you wish to communicate with (usually these are all other hosts running the MSDT CLI server).
6. Enter **`d`** when you have entered the IP addresses of all hosts.

note This installation process creates an **access.dat** file in the **/etc/opt/sanmgr/hostagent/config** directory with the IP addresses you entered. You can edit this file at any time to change the list of hosts you wish to communicate with.

7. Enter the IP address of the local LAN interface used for outbound communication:
   - If the system has more than one network adapter (is multihomed), enter the IP address of the network interface card to be used by the MSDT CLI server.
   - If the system has only one network interface, enter that IP address.

note This installation process creates the **commIpAddr.txt** file with the IP address you entered in the **etc/sanmgr/hostagent/config** directory of your HP-UX system. You can edit this file at any time to change the designated IP address.

8. Repeat these steps for each host machine.

option 2: install from download

1. Download latest tar file from HP Storage tools web site from **http://storagetools.lvld.hp.com**
2. Create a temporary directory on the server.
3. Use ftp to download the tar file to the temporary directory you created.
4. Extract the tar file:

   # tar -xvf *filename*.**tar**
5. Type the following command to start the installation script:

   # ./install_hpux_msdt

   The installation script launches, checks for the existence of SAM Host Agent version 3.0 (see above), and prompts you for information.
6. Enter, individually, the IP addresses of each host running the MSDT CLI server that you wish to communicate with (usually these are all other hosts running the MSDT CLI server).
7. Enter **d** when you have entered the IP addresses of all hosts.

note This installation process creates an **access.dat** file in the **/etc/opt/sanmgr/hostagent/config** directory with the IP addresses you entered. You can edit this file at any time to change the list of hosts you wish to communicate with.

8. Enter the IP address of the local LAN interface used for outbound communication:
   - If the system has more than one network adapter (is multihomed), enter the IP address of the network interface card to be used by the MSDT CLI server.
   - If the system has only one network interface, enter that IP address.

note This installation process creates the **commIpAddr.txt** file with the IP address you entered in the **etc/sanmgr/hostagent/config** directory of your HP-UX system. You can edit this file at any time to change the designated IP address.

9. Repeat these steps for each host machine.

installing the tools with a firewall

To install the MSDT Tools with a firewall:

1. Follow the steps in “installing the tools” on page 7 to install the Tools.
2. Use the **msdtstop** command to stop the Tools.
3. Configure the port numbers (see below).
4. Configure the firewall (see below).
5. Use the **msdtstart** command to restart the Tools.

configuring port numbers

The Tools server is preconfigured to use the port numbers listed in the **StartHACfg.prp** and **ShutdownHACfg.prp** files. The **StartHACfg.prp** and **ShutdownHACfg.prp** files are copied to the **ServiceManagerCfg.prp** file prior to startup and shutdown, respectively, of the HostAgent service. If necessary, open the **StartHACfg.prp** and **ShutdownHACfg.prp** files and edit the port numbers.

If the **ServiceManagerCfg.prp** file already exists, then you must edit the **ServiceManagerCfg.prp** file instead of the **StartHACfg.prp** file.

host agent software

The **ServiceManagerCfg.prp** file for the Host Agent software uses the port numbers listed in the following table:

host agent software port numbers

configuration entry	port number	protocol
RMI_PORT	64301, 64320	TCP
QDISCOVERY/FINDER/PORT	64311, 64330	UDP
QDISCOVERY/FINDER/UNICAST_AGENT/PORT	64300	UDP
QDISCOVERY/RESPONDER/UNICAST_AGENT/PORT	64300	UDP

using customized port numbers

The port numbers in the supplied files are recommended values. If you want to use a customized set of port numbers, they must be:

- Valid TCP and UDP port numbers between 1 and 65535. A value of zero results in a random available port number, which is not valid for firewall configurations.
- Available ports on the host where the Tools are running.

configuring the firewall

When Storage Area Manager hosts are separated by a firewall, you must configure each component to restrict its communication to specific ports. You must configure the firewall to open the UDP and TCP ports that are used by the HostAgent.

note For specific instructions on configuring a firewall, refer to the documentation provided by the firewall supplier.

The table below lists the inbound (listening) port numbers for the HostAgent software that must be opened on the surrounding firewall. Open these ports through the firewall to the specified component.

For example, a firewall exists between the local MSDT CLI server (where the command is invoked) and the remote MSDT CLI server host (where the command is executed). The HostAgent software that contains the remote MSDT CLI server is listening on TCP port 64301. Therefore, port 64301 must be open for communication from the local MSDT CLI server's subnet to the remote server host subnet.

configuring firewall ports

component	UDP ports	TCP ports
HostAgent software	2715 64300 to 64309 64311, 64330	64301, 64320

The table also shows the values used in the firewall configuration files provided in the installation package. If you use custom port values (see page 9), the firewall port settings will be different. All HostAgent components use UDP port 2715 for multicast at 228.5.6.7. If multicast is not allowed to pass through the firewall, you do not have to open these ports.

configuring firewall support

Firewall implementations must support and use TCP state tracking (also called connection tracking). The Tools do not accept firewall implementations that use network address translation (NAT).

note If HP OpenView StorageArea Manager (SAM) is installed on your system, use the SAM user guide to configure firewall support.

uninstalling the tools

For HP-UX systems:

1. Log in as **root**.
2. Enter **# cd /etc/opt/hpmsdt/install**
3. Enter **# ./uninstall_hpux_msdt**
4. Repeat these steps on each system where you want to uninstall the Tools.

3 Configuration File

The configuration file contains detailed information on the applications, hosts, and device groups in the MSDT Solution, and enhances the functionality of the Tools CLI.

The configuration file defines:

- Meta-data for the application
- The device groups in the application

The configuration file enables you to execute RM-like commands on the local host by only specifying the appropriate device group name and the actions for the command. The commands then attempt to execute the appropriate commands on all the hosts and instances defined in the configuration file, and return the output of the actual RM commands.

Using the configuration file, you associate hosts with a device group name so that you can execute a command on one of the many hosts associated with that device group, as opposed to a specific destination host in the configuration file. The first successful execution of the command returns a success code. If the command fails to execute for any reason, the CLI returns a failure code. This relieves you from having to know the exact host and instance that forms that part of the configuration.

The installation process installs a sample configuration file named **`msdt.cfg`** in the **`/etc/opt/hpmsdt/conf/sample`** directory. Copy this sample file to **`/etc/opt/hpmsdt/conf/msdt.cfg`**. Use a text editor to enter the appropriate information for each application that is managed in your MSDT Solution. All hosts defined in the configuration file must have the Tools installed and have access permissions set to communicate with all of the other hosts in the solution.

The following sections explain the syntax and parameters of the configuration file:

mgr_hosts

This section lists the hosts running the management scripts. You do not need to include hosts that will run applications or manage device groups.

These are normally hosts that do not have direct access to any of the storage arrays, and are used only for management of the application. These hosts will maintain common log files for events in the configuration. If no such host exists, leave this part empty.

```
<mgr_hosts>={host1,host2,host3...}
```

application names

Define your application names in this section. Follow these rules:

- The name must be unique within the configuration file.
- The name must be one word.
- The name can contain characters, numbers, and underscores.
- Although there is no limit to the length of the application name, it is recommended that you keep the application name to fewer than 20 characters.
- Every application in the solution requires a separate application section.

```
<application>=application_name
```

data centers

In this section, define your data centers that will run the application. Follow these rules:

- Include one entry for each data center (with or without application hosts).
- If a data center does not have any hosts to run the application, define the data center with an empty list of hosts.
- A host can only be listed in one data center.
- Data center and host names must be one word.
- Data center and host names can contain characters, numbers, and underscores.
- Data center names should be less than 20 characters.

The order in which you list the data centers is important. The default definition lists the first data center as the primary data center , the second is the target for the synchronous CA replication, and the third is the target for the long-distance CA data center.

```
<datacenter>= datacenter1 { <apphost>=hostA1, hostA2, hostA3, hostA4 }
<datacenter>= datacenter2 { <apphost>=hostB1, hostB2, hostB3 }
<datacenter>= datacenter3 { <apphost>=hostC1 }
```

device groups

In this section, list the device groups and the groups that will manage them. Follow these rules:

- Each device group must have a name (recommended not to exceed 15 characters).
- The device group name must be unique within the configuration file.
- The device group name must be one word.
- The device group name can contain characters, numbers and underscores (_).
- Each device group must have an array_type. The default is “XP.” Currently, only XP is supported.
- Each device group must have a mirror_type. Currently, only CA and BC are supported.

```
<device_group> = group_name {
   <array_type> = XP
   <mirror_type> = CA
   <mirror_level> = CA_1
   <left> = datacenter1 {
      <host> = hostA1 {<inst>=0}
      <host> = hostA2 {<inst>=0, 1, 2}
      <host> = hostA5 {<inst>=0,2}
   }
   <right> = datacenter2 {
      <host> = hostB1 {<inst>=0, 1}
      <host> = hostB2 {<inst>=1, 2}
```

mirror levels

You can use a mirror level for each device group. Mirror levels determine the relationships between device groups in the MSDT Solution. The mirror level must be unique to the device group within the application (in other words, for each application, there can only be one device group with a particular mirror level). Valid mirror levels are:

- CA_1 — The sync CA device group between data centers 1 and 2
- BC_1 — The BC group that will make the point-in-time image (at data center 1 or 2)
- CA_2 — The long-distance CA link between data centers 2 and 3 (or 1 and 3)
- BC_2 — The remote BC copy at data center 3

If no mirror level is defined, the default value of "unknown" will be assigned to that group.

left/right sections

Each device group must have a left *or* right section defined, but it is best to define both. The default definition lists the left side as the P-vol of the device group and the right side as the S-vol. The specific function of the devices on either the left or right side can change at any time during the lifetime of the solution. Follow these rules:

- Each left and right section must have a data center name defined. Names should match the one of the names defined in the data centers section.
- Each left and right section must have at least one host defined.
- Each host must have at least one instance defined.

user-defined values

User-defined values are any values that you may require in additional scripting. To define these values, you may use this section or create a separate configuration file.

Template scripts use these parameters to determine specific operations. The syntax of these parameters are not checked by the verification process of the Tools commands, and are only checked by the template scripts.

```
<user_defines> {
msdt_type=4
cycle_type=0
pre_options="-a all -shutdown"
post_options="-start"
get_all_status=0
verbose=0
Force=0
}
```

msdt type

The **`msdt_type`** parameter defines the type of multi site configuration implemented. If all four device groups are implemented, you must set the **`msdt_type`** parameter to 4, indicating four links. If the optional BC device group on Site 3 is not used (HP highly recommends using this device group), then set the **`msdt_type`** to 3, which indicates that only three links exist. These are the only values allowed for this parameter.

cycle type

The **cycle_type** defines the required status of the device pairs for the cycle process to start. **cycle_type=0** indicates that all device groups except the CA_1 device group must be in suspend status in order to initiate a cycle process.

cycle_type=1 indicates that the BC_1 device group must be in pair status and the CA_2 and BC_2 device groups must be in suspend status in order to initiate the cycling of data.

pre and post options

The **pre_options** define the optional parameters to execute the **pre_exec** script before suspending the BC_1 device. You can use this script to prepare the application for the creation of the point-in-time image during the cycle process.

The **post_options** define the optional parameters to pass to the **post_exec** script that runs after the creation of the point-in-time image in order to return the application to normal operations.

Set the **get_all_status=***0|1* variable to 1 to force the **cycle_pair** script to collect all device group statuses after each event in the cycle process. This is more recourse intensive, but ensures that every device change is detected, especially if a long time laps occur between events in the cycle process. The default value 0 forces the script to collect only the new status of the device group it is currently operating on.

cycle pair

Set the **verbose=***0|1* variable to 1 to force the **cycle_pair** script to display all device group statuses after each event in the cycle process. The default 0 only displays the device group status at the start of the cycle pair process.

The default **Force=0** value forces the **cycle_pair** script to obtain a exit code from every host that is able to run the application in order to ensure that there is only one host actually running the application. If one of the defined hosts is not reached to determine if the application is running on that host, the script will fail to complete the cycle process. When the **force** parameter is set to 1, the **cycle_pair** script will continue to cycle the data even if a host does not report the application status. You run the risk that the host running the application cannot communicate with the management hosts, and therefore the script assumes the application is not running and does not execute the **pre_exec** and **post_exec** scripts. You should only set this value if one of the hosts in the configuration is disabled.

sample configuration file

The following is an example of the formats and parameters of a valid configuration file:

```
<mgr_hosts>={sanmgr1,sanmgr2}
<application>=oracle {
   <datacenter>=DC_one { <apphost>=alpha108, alpha109 }
   <datacenter>=DC_two { <apphost>=alpha155, alpha156 }
   <datacenter>=DC_three { <apphost>=alpha154 }

   <device_group> = ca1_oracle {
      <array_type> = XP
      <mirror_type> = CA
      <mirror_level> = CA_1
      <left> = DC_one {
         <host> = alpha108 {<inst>=0,1}
         <host> = alpha109 {<inst>=0,1}
      }
      <right> = DC_two {
         <host> = alpha155 {<inst>=0,1}
         <host> = alpha156 {<inst>=0,1}
      }
   }
   <device_group> = bc1_oracle {
      <array_type> = XP
      <mirror_type> = BC
      <mirror_level> = BC_1
      <left>=DC_two {
         <host> = alpha155 {<inst>=2}
         <host> = alpha156 {<inst>=2}
      }
      <right>=DC_two {
         <host> = alpha155 {<inst>=3}
         <host> = alpha156 {<inst>=3}
      }
   }
   <device_group> = ca2_oracle {
      <array_type> = XP
      <mirror_type> = CA
      <mirror_level> = CA_2
      <left>=DC_two {
         <host> = alpha155 {<inst>=4}
         <host> = alpha156 {<inst>=4}
      }
      <right>=DC_three {
```

```
         <host> = alpha154 {<inst>=5}
      }
   }
   <device_group> = bc2_oracle {
      <array_type> = XP
      <mirror_type> = BC
      <mirror_level> = BC_2
      <left>=DC_three {
         <host> = alpha154 {<inst>=6}
      }
      <right>=DC_three {
         <host> = alpha154 {<inst>=7}
      }
   }
   <user_defines> {
      msdt_type=4
      cycle_type=0
      pre_options="-a all -shutdown"
      post_options="-start"
      get_all_status=0
      verbose=0
      Force=0
   }
}
```

4 Commands

The Tools commands allow you to initiate RM operations at several hosts from a remote server.

syntax

The commands, entered in the MSDT CLI, use a unique command language with the following syntax:

syntax description

argument	description
Value	Indicates that you must supply a value
[ ]	Indicates an optional parameter
[[ ]]	Indicates an optional parameter that is only valid when used with another optional parameter

The following table contains the complete list of commands, which are described in detail throughout this chapter:

commands

command	syntax and parameters
displaypair	**displaypair -g** *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-l] [-fc] [-fx] [-fd] [-f[x][c][d]] [-m (cas|all)] [-CLI]**
statuspair	**statuspair -g** *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-P|-S] [-s[s]] [-c] [-nomsg]**
suspendpair	**suspendpair -g** *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-r|-rw] [-l] [-c** *size***] [-nomsg]**
resumepair	**resumepair -g** *group name* **[-h** *host name* **[-i** *instance number***] [-v] [-swaps|-swapp]] [-c** *size***] [-l] [-nomsg] [-restore]**
eventwaitpair	**eventwaitpair -g** *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-s smpl] [-s copy] [-s pair] [-s psus] [-s psue(psuse)] [-t** *time_out***] [-l] [-nomsg] [-nowait]**
swappair	**swappair -g** *group name* **-h** *host name* **[-i** *instance number***] [-v][-S] [-l] [-t** *time_out***] [-nomsg]**
createpair	**createpair -g** *group name* **-h** *host name* **[-i** *instance number***] [-v] [-f** *fence_level* **[CTGID]] -vl|-vr [-c** *size***] [-nocopy] [-split] [-m noread] [-m cyl] [-m trk] [-m dif] [-m inc] [-m grp [GID]]**
deletepair	**deletepair -g** *group name* **[-h** *host name* **[-i** *instance number***]] -S|-R|-P [-v] [-l] [-nomsg]**
find_app	**find_app -a** *application_name* **[-v] [-o "***options***"]**
pre_exec	**pre_exec -a** *application_name* **-h** *host name* **[-o "***options***"]**
post_exec	**post_exec -a** *application_name* **-h** *host name* **[-o "***options***"]**
dist_conf	**dist_conf [***config file***]**
dist_log_msg	**dist_log_msg –a** *application_name* **-o "***message***"**
msdtverify	**msdtverify [***config file***]**
msdtversion	**msdtversion**
msdtinfo	**msdtinfo [-f** *config file***]** *arguments*

command	syntax and parameters
msdtstart	**msdtstart**
msdtstop	**msdtstop**
msdtrestart	**msdtrestart**
msdtcmd	**msdtcmd**

return values

Return values from distributed meta-commands tell you whether a command executed successfully or failed. The value of the code depends on the parameters of the command. Each command has its own return values.

Distributed commands attempt to execute on multiple hosts until they find a host that successfully runs the command. Upon successful execution, the command exits and returns a success value. Most commands require only one host to successfully execute the command and exit with a success value. However, some commands require successful execution on two or more hosts in order to exit with a success code. If the command fails to execute successfully on all hosts, the command exits and returns a failure code. Additionally, if a command cannot complete its required actions due to connection failure, remote execution failure, or other event, the command returns a failure value.

displaying detail pair information — displaypair

The **displaypair** command executes the RM **pairdisplay** command for a specified device group. The command searches the configuration file for:

- A list of hosts that can manage the device group
- The number of the RM instance on each of these hosts
- The mirror type for this device group

```
displaypair -g group name [-h host_name [-i instance number]] [-v] [-l] [-fc] [-fx] [-fd]
[-f[x][c][d]] [-m (cas|all)] [-CLI]
```

parameters

You can use the following parameters with the **displaypair** command:

displaypair parameters

parameter	description
-g *group name*	Required. This parameter specifies on what device group to execute the command. The *group name* value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[-fc] [-fx] [-fd] **[-f[x][c][d]]** **[-m (cas|all)] [-CLI]**	Optional. These parameters are passed directly to the **pairdisplay** command to control the output of this command. For more detail on these parameters, see the RM user manual.
[-l]	Optional. Forces the **pairdisplay** command to only query devices local to the system it is executing on. This parameter attempts to collect local information for the P-vol and S-vol of the pair from two different hosts.

output

The **displaypair** command outputs:

- The Data center on which the command was executed
- The host name of the host that executed the command
- The RM instance used for the command
- The standard output of the **pairdisplay** command
- The return value of the RM command

sample output: –l parameter not used

This example shows the output for a **displaypair** command that does not include the **-l** parameter. The example shows the host and instance used to produce the output, as well as the actual output of the **pairdisplay** command.

Since this command does not include the **-l** parameter, the output lists only the *first* host to successfully execute the **pairdisplay** command. It also shows both P-vol and S-vol information as viewed from this host.

```
# displaypair -g vg_fs5
Data Center : DC_two
Host        : alpha108
Instance    : 0
Type        : CA
Level       : CA_1
Command     : /usr/bin/pairdisplay -g vg_fs5
Group  PairVol(L/R) (Port#,TID,LU), Seq#,  LDEV#. P/S,  Status, Fence,  Seq#,  P-LDEV# M
vg_fs5  pv_fs5(L)   (CL2-L, 1, 6)   20035  896..  S-VOL  PAIR   NEVER,  -----  708     -
vg_fs5  pv_fs5(R)   (CL1-D, 1, 6)   20030  708..  P-VOL  PAIR   NEVER,  20035  896     -
```

sample output: -v parameter used

This example uses the same command as the first example, but includes the **-v** parameter; therefore the output displays *all* attempts to execute the command **and** all errors encountered during execution. The output indicates that:

- Host alpha154, instance 6 does not have a group called vg_fs5 defined
- Host alpha108 instance 5 failed to start
- Host alpha108 instance 0 successfully executed the command

```
# displaypair -g vg_fs5 -v
Data Center  : DC_two
Host         : alpha154
Instance     : 6
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairdisplay -g vg_fs5
pairdisplay  : [EX_ENOGRP] No such group
Refer to the command log(/HORCM/log6/horcc_alpha154.log) for details.

Data Center  : DC_two
Host         : alpha108
Raid Manager instance: 5 failed to start

Data Center  : DC_two
Host         : alpha108
Instance     : 0
Type         : CA
```

```
Level        : CA_1
Command      : /usr/bin/pairdisplay -g vg_fs5
Group  PairVol(L/R) (Port#,TID,LU), Seq#,  LDEV#. P/S,  Status, Fence,  Seq#,  P-LDEV# M
vg_fs5   pv_fs5(L)  (CL2-L, 1, 6)   20035  896..  S-VOL  PAIR   NEVER,  -----  708     -
vg_fs5   pv_fs5(R)  (CL1-D, 1, 6)   20030  708..  P-VOL  PAIR   NEVER,  20035  896     -
```

sample output: -l option used

This example shows the output for a **displaypair** command that includes the **-l** parameter. The **-l** option forces the **displaypair** command to execute a **pairdisplay** for both the left and right side of the device pair. Both hosts used to collect the information are shown with their local devices.

```
# displaypair -g vg_fs5 -l
Data Center  : DC_two
Host         : alpha108
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairdisplay -g vg_fs5 -l
Group  PairVol(L/R) (Port#,TID,LU), Seq#,  LDEV#. P/S,  Status, Fence,  Seq#,  P-LDEV# M
vg_fs5   pv_fs5(L)  (CL2-L, 1, 6)   20035  896..  S-VOL  PAIR   NEVER,  -----  708     -

Data Center  : DC_two
Host         : alpha155
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairdisplay -g vg_fs5 -l
Group  PairVol(L/R) (Port#,TID,LU), Seq#,  LDEV#. P/S,  Status, Fence,  Seq#,  P-LDEV# M
vg_fs5   pv_fs5(L)  (CL1-D, 1, 6)   20030  708..  P-VOL  PAIR   NEVER,  20035  896     -
```

sample output: specific host/instance defined

This example shows the output from a **displaypair** command that defines a specific host and instance. The command also includes the **-fcx** and **-CLI** parameters to manipulate the output of the **pairdisplay** command.

```
# displaypair -g vg_fs5 -v -fcx -CLI -h alpha109 -i 1
Data Center  : DC_two
Host         : alpha109
Instance     : 1
Type         : CA
Level        : CA_1
Command     : /usr/bin/pairdisplay -g vg_fs5 -fcx -CLI
Group  PairVol L/R   Port#  TID LU   Seq#   LDEV#  P/S   Status  Fence     %    P-LDEV# M
vg_fs5 pv_fs5 L      CL2-L  1   6    20035  380    S-VOL  PAIR   NEVER    100   2c4     -
vg_fs5 pv_fs5 R      CL1-D  1   6    20030  2c4    P-VOL  PAIR   NEVER    100   380     -
```

return values

displaypair return values

Normal termination:	0: Indicates successful execution of the command
Abnormal termination	250: Invalid configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

displaying pair status — statuspair

The **statuspair** command executes the RM **pairvolchk** command on a given device pair. Unless there is an abnormal termination, the **pairvolchk** command determines the status of the pair's primary and secondary volume sides and produces a return code. Depending on the parameters you set in the **statuspair** command, the command either returns the same value as the **pairvolchk** command, or simply tells you that the RM command executed successfully.

statuspair -g *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-P|-S] [-s[s]] [-c] [-nomsg]**

parameters

You can use the following parameters with the **statuspair** command:

statuspair parameters

parameter	description
-g *group name*	Required. This parameter enables you to specify what device group to execute the command. The group name value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[-P|S]	Optional. If not specified, the command will just display the status of the P-vol and S-vol in two separate lines and exit with a code 0. you can specify the **[-P|S]** option to restrict the output of the command to the primary or secondary volume status. If used in this manner, you will call the command once with the **-P** option and again with the **-S** option. Since it is impossible to know if the left or right definition of the device group will be the P-vol, you must issue the command on the first host in the list, evaluate the return code, and issue the command on the other side of the device group if this is not the appropriate side. When using the **-P** or **-S** options the output should only contain the valid output for that volume and not any failure output. if the first attempt result in P-vol information and the command is looking for S-vol information then the command should not print this output. You cannot use the **[-P|S]** option with the **[-h** *host name***]** option.
[-s[s]] **[-c]** **[-nomsg]**	Use these optional parameters to change the output of the **pairvolchk** command. The **-s[s]** option enables you to acquire the fine granularity volume state (e.g. pvol_psus) of a volume. If you do not specify the **-s[s]**, the generic volume state (e.g. P-vol) is reported. The **[-c]** option enables you to obtain the status of the remote side of the pair from the point of view of the host executing the command. Users do not often execute this command, and the output might be very misleading because the host and status information displayed in the output will not match. Use the **-nomsg** to suppress messages, and when you only want the return code of the **pairvolchk** command.

output

The output of the **statuspair** command outputs indicates the:

- Data center
- Host name
- RM instance
- Mirror type
- Mirror level
- Command used to execute the remote command
- Actual output of the remote command.

sample output: no –P|S parameter or specific host used

This example shows the output of a **statuspair** command that does not include the **–P|S** parameter or specify the **[–h** *host name***]**. The output provides the status of both the primary and secondary volume sides of the pair and exits with a 0 return value.

note Because the left and right primary and secondary volume sides of the pair report different return values, it is impossible to return the RM return values.

```
# statuspair -g vg_fs5
Data Center  : DC_two
Host         : alpha108
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairvolchk -g vg_fs5
pairvolchk   : Volstat is S-VOL.[status = PAIR fence = NEVER]

Data Center  : DC_two
Host         : alpha155
Instance      : 0
Type          : CA
Level         : CA_1
Command       : /usr/bin/pairvolchk -g vg_fs5
pairvolchk    : Volstat is P-VOL.[status = PAIR fence = NEVER]
```

sample output: –P parameter specified

This example shows the output of a **statuspair** command that includes the **-P** parameter. This output gives the status of the primary volume side of the pair only. The exit value of the **statuspair** command is the same as the **pairvolchk** command.

note When using the **-P** or**-S** parameters, the output only contains a success code for that volume and no failure output.

```
# statuspair -g vg_fs5 -P
Data Center  : DC_two
Host         : alpha155
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairvolchk -g vg_fs5
Pairvolchk   : Volstat is P-VOL.[status = PAIR fence = NEVER]
```

sample output:–S parameter specified

This example shows the output of a **statuspair** command that includes the **-S** parameter. This output gives the status of secondary volume side of the pair only. The exit code of the **statuspair** command is the same as the **pairvolchk** command.

note In this example, the command included the **-ss** parameter to show the finer status level, and the return code from the RM **pairvolchk** is 33.

```
# statuspair -g vg_fs5 -S -ss
Data Center  : DC_two
Host         : alpha108
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairvolchk -g vg_fs5 -ss
Pairvolchk   : Volstat is S-VOL.[status = PAIR fence = NEVER]
```

return values

–P|S options not specified

condition	value
Normal termination	0: Indicates successful execution of the command on least one side of the device group.
Abnormal termination	250: Invalid configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

–P|S or –h host name and –i instance number specified

condition	value
Normal termination	1: The volume attribute is “SMPL” 2: The volume attribute is “P-vol” 3: The volume attribute is “S-vol” 11: The status is “SMPL” 22: The status is “PVOL_COPY” or “PVOL_RCPY” 23: The status is “PVOL_PAIR” 24: The status is “PVOL_PSUS” 25: The status is “PVOL_PSUE” 26: The status is “PVOL_PDUB” 29: The status is “PVOL_INCSTG” Inconsistent status in the group 32: The status is “SVOL_COPY” or “SVOL_RCPY” 33: The status is “SVOL_PAIR” 34: The status is “SVOL_PSUS” 35: The status is “SVOL_PSUE” 36: The status is “SVOL_PDUB” 39: The status is “SVOL_INCSTG” Inconsistent status in the group 42: The status is “PVOL_COPY” 43: The status is “PVOL_PAIR” 44: The status is “PVOL_PSUS” 45: The status is “PVOL_PSUE” 46: The status is “PVOL_PDUB” 47: The status is “PVOL_PFUL” 48: The status is “PVOL_PFUS” 52: The status is “SVOL_COPY” 53: The status is “SVOL_PAIR” 54: The status is “SVOL_PSUS” 55: The status is “SVOL_PSUE” 56: The status is “SVOL_PDUB” 57: The status is “SVOL_PFUL” 58: The status is “SVOL_PFUS”
Abnormal termination	250: Invalid configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

suspending mirror operations — suspendpair

The **suspendpair** command executes the **pairsplit** command for a given pair. The **pairsplit** command:

- Suspends the mirror operation between primary and secondary devices
- Read/write enables the secondary devices (optional)

suspendpair -g *group name* **[-h** *host name* **[-i** *instance number***]] [-v] [-r|-rw] [-l] [-c** *size***] [-nomsg]**

parameters

You can use the following parameters with the **suspendpair** command:

suspendpair parameters

parameter	description
-g *group name*	Required. This parameter specifies what device group to execute the command on. The group name value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[-r|rw] **[-c** *size***]** **[-nomsg]**	Optional. Enables you to manipulate the operation of the **pairsplit** command. The **-r** or **-rw** option specifies if the secondary volume must be read only or read/write enabled.
[-l]	Optional. The **-l** option ensures successful operation even if the remote RM instance is not available. This option will only work if the P-vol is the local device. If the **-l** option is used on the S-vol, the return value of the **pairsplit** command is 222. .

output

The output of the **suspendpair** command lists the:

- Data center
- Host name
- RM instance
- Mirror type
- Mirror level command used in executing the remote command
- Output of the remote command

sample output: suspendpair

This example shows the output of the **`suspendpair`** command. The output gives the host and instance that executed the command as well as the status of the command.

```
# suspendpair -g vg_fs5
Data Center      : DC_two
Host             : alpha108
Instance         : 0
Type             : CA
Level            : CA_1
Command          : /usr/bin/pairsplit -g vg_fs5
suspendpair completed successfully
```

return values

suspendpair return values

condition	value
Normal termination	0: Indicates successful execution of the command on least one host in the configuration.
Abnormal termination	250: Invalid configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

resuming mirror operations — resumepair

The **resumepair** command executes the RM **pairresync** command for a given pair. The **pairresync** command resumes the mirror operation between a primary and secondary device. You can use the **–swapp** or **-swaps** parameters in this command to complete a previously failed takeover command. This command also enables you to restore BC data from the secondary volume to the primary volume.

resumepair -g *group name* **[-h** *host name* **[-i** *instance number***] [-v] [-swaps|-swapp]] [-c** *size***] [-l] [-nomsg] [-restore]**

parameters

You can use the following parameters with the **resumepair** command:

resumepair parameters

parameter	description
-g *group name*	Required. This parameter specifies what device group to execute the command on. The group name value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[–swapp or –swaps]	Optional. Use this parameter only if you specified a specific host to execute this command on. The **-swapp** option must run on the P-Vol side of the pair and the **–swaps** on the S-Vol side of the pair. You will typically use the **statuspair** command to find the host that is currently controlling the P-Vol or S-Vol, and then use the **resumepair** pair to execute the appropriate command on the correct host.
[-c *size***]** **[–l]** **[–nomsg]** **[-restore]**	Optional. Enables you to manipulate the operation of the actual "pairresync" command. The **-c** *size* command indicates the number of tracks that must be copied and can range from 1 to 15. The **–l** option is useful if the remote RM instance is not running. Use the **–restore** only for BC devices. The **–restore** parameter executes a data restore from the secondary volume to the primary volume.

output

The output of the **resumepair** command indicates the:

- Data center
- *Host name*
- RM instance
- Mirror type
- Mirror level
- Command used in executing the remote command
- Output of the remote command

sample output: -v parameter specified

This example shows the output of the **resumepair** command with the **-v** parameter specified. After a number of failures, the **pairresync** successfully executes, and returns a 0 value.

```
# resumepair -g vg_fs5 -v
Data Center  : DC_two
Host         : alpha154
Instance     : 6
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairresync -g vg_fs5
pairresync: [EX_ENOGRP] No such group
Refer to the command log(/HORCM/log6/horcc_alpha154.log) for details.

Data Center  : DC_two
Host         : alpha108
Raid Manager instance: 5 failed to start

Data Center  : DC_two
Host         : alpha108
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/pairresync -g vg_fs5
Resumepair completed successfully
```

return values

resumepair return values

condition	value
Normal termination	0: Resume was successful
Abnormal termination	250: Invalid Configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

waiting for specific events — eventwaitpair

The **eventwaitpair** command performs the RM **pairevtwait** command for a given pair. Use the **pairevtwait** command to constantly monitor a device group until it reaches a specified status. Once it reaches this status, the command exits with a normal exit value. If the timeout value is reached before the desired status change, then the command exits with an abnormal exit value. This command is useful because it immediately responds to status changes, opposed to polling the device at regular intervals to detect changes.

note Because this command waits for the timeout to complete, execution time can be very long. The command uses the communication channel between hosts, blocking the port and preventing other communication. You should use short timeout values, and retry the command on regular intervals until it reaches the desired status.

```
eventwaitpair -g group name [-h host name [-i instance number]] [-v] [-s smpl] [-s copy]
[-s pair] [-s psus] [-s psue(psuse)] [-t time_out] [-l] [-nomsg] [-nowait]
```

parameters

You can use the following parameters with the **resumepair** command:

resumepair parameters

parameter	description
-g *group name*	Required. This parameter enables you to specify what device group to execute the command. The group name value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[-s smpl] **[-s copy]** **[-s pair]** **[-s psus]** **[-s psue]** **[-t** *time_out***]** **[-l]** **[-nomsg]** **[-nowait]**	Optional. These parameters enable you to manipulate the operation of the RM **pairevtwait** command. The **-s** *status* command identifies the status that you want RM to respond to. Since the RM only responds to the specified status, you should specify every possible status that can be triggered so you are notified of general status changes. The **-t** *timeout* option specifies the time in seconds to wait for a status change. The **-l** option is useful if the remote RM instance is not running. The **-nowait** option returns immediately with an exit code indicating the current pair status. If you specify the **-nowait** option, the **-t** option will be ignored.

output

The output of the **`eventwaitpair`** command shows the:

- Data center
- Host name
- RM instance
- Mirror type
- Mirror level
- Command used to execute the remote command
- Output of the remote command

sample output: eventwaitpair

This example shows the output of the **`eventwaitpair`** command. The output gives the host and instance that executed the command as well as the command status.

```
# eventwaitpair -g vg_fs5 -s pair -t 30
Data Center : DC_two
Host        : alpha108
Instance    : 0
Type        : CA
Level       : CA_1
Command     : /usr/bin/pairevtwait -g vg_fs5 -s pair -t 30
pairevtwait : Wait status done.
```

return values

eventwaitpair return values

condition	value
Normal termination	0: Indicates successful execution of the command on at least one host in the configuration
Normal with **`-nowait`** parameter	1: The status is “SMPL” 2: The status is “COPY” or “RCPY” 3: The status is “PAIR” 4: The status is “PSUS” 5: The status is “PSUE”
Abnormal termination	250: Invalid Configuration file 251: Config file not found 252: Command time out 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

swapping primary and secondary devices — swappair

The **swappair** command performs the RM **horctakeover** command for a given pair. The **horctakeover** command swaps the personalities of CA devices and attempts to resynchronize the device group in the opposite direction. The action and return codes of this command depend on local device status; for this reason users must specify the host name and any optional instance numbers.

This command is only applicable to CA device groups. If you attempt it on BC device groups, the command will fail.

swappair -g *group name* **-h** *host name* **[-i** *instance number***] [-v][-S] [-l] [-t** *time_out***] [-nomsg]**

parameters

You can use the following parameters with the **swappair** command:

swappair parameters

parameter	description
-g *group name*	Required. This parameter enables you to specify what device group to execute the command. The group name value must match a device name entry in the configuration file.
-h *host name*	Required. You must use this parameter to ensure that the command executes on the correct host. The host name must only appear on one side of the device group definition. If the host name appears on both sides of the device group definition, you must include the **-i** *instance number* parameter to uniquely identify the swap direction. If you use supply a host name without the **–i** *instance number* parameter, that host name must be defined in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-i *instance number***]**	Optional. Executes the command on a specific instance. If you do not specify the **–i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text.
[-S] **[-l]** **[-t** *timeout***]** **[-nomsg]**	Optional. These parameters enable you to manipulate the operation of the "horctakeover" command. The **–S** option is used to select and execute a S-vol takeover. With this option the local device have to be a S-vol. The –**t** *timeout* option specifies the time in seconds to wait for P-Vol to S-Vol delta data resynchronization to complete in an Async CA device group. The **–t** option is ignored in a sync CA device group. The **–l** option is useful if the remote RM instance is not running.

output

The output of the **swappair** command shows the data center, host name RM instance, mirror type, mirror level, and command used in executing the remote command as well as the actual output of the remote command.

sample output: swappair

This example shows the output of the **`swappair`** command. The output gives the host and instance that executed the command as well as the command status.

```
# swappair -g vg_fs5 -h alpha109
Data Center  : DC_two
Host         : alpha109
Instance     : 0
Type         : CA
Level        : CA_1
Command      : /usr/bin/horctakeover -g vg_fs5
horctakeover : Swap-Takeover done.
```

return values

swappair return values

Condition	value
Normal termination	0: Nop-takeover (no operation) 1: Swap takeover was successfully executed 2: SVOL takeover was successfully executed 3: P-vol-SMPL-takeover was successfully executed 4: P-vol-PSUE-takeover was successfully executed 5: S-vol-SUSE-takeover was successfully executed
Abnormal termination	250: Invalid Configuration file 251: Config file not found 252: Command time out 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

creating device pairs — createpair

The **createpair** command executes the RM **paircreate** command for a given pair. The **paircreate** command:

- Creates a new relationship between two devices
- Starts the copy process from the primary to the secondary device

note The direction of pair creation depends on the parameters you supply and the host the command is executed on. For this reason, you must specify a host name for this command, in addition to any optional instance numbers.

```
createpair -g group name -h host name [-i instance number] [-v] [-f fence level [CTGID]] -vl|-vr
[-c size] [-nocopy] [-split] [-m noread] [-m cyl] [-m trk] [-m dif] [-m inc] [-m grp [GID]]
```

parameters

You can use the following parameters with the **createpair** command:

createpair parameters

Parameter	Description
-g *group name*	Required. This parameter enables you to specify what device group to execute the command. The group name value must match a device name entry in the configuration file.
-h *host name*	Required. You must use this parameter to ensure that the command executes on the correct host. The host name must only appear on one side of the device group definition. If the host name appears on both sides of the device group definition, you must include the **-i** *instance number* parameter to uniquely identify the direction for the **paircreate**. If you use supply a host name without the **-i** *instance number* parameter, that host name must be defined in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
[-i *instance number***]**	Optional. Executes the command on a specific instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file.
[-f *fence* **[CTGID]]** **[-vl|-vr]** **[-c** *size***]** **[-nocopy]** **[-split]** **[-m** *mode***]**	Optional. These parameters enable you to manipulate the operation of the actual **paircreate** command. The **-f** *fence* **[CTGID]** is only applicable for CA device groups and defines the fence level for the device group and optionally the consistency group id for Async CA. The **-vl** and **-vr** option determine if the local device (**-vl**) or remote device (**-vr**), in relationship to the instance issuing the command, will be the Primary device for the device pair. The **-c** *size* option specifies the number of tracks to copy at one time and can be used to control the load on the CA link. The **-nocopy** option is used for CA only and indicate that the pair was in "pair" status before and none of the information on the device must be copied again. The **-split** option is for BC only and allows for the Secondary volume to be available immediately after the create process with a full background copy still pending. The **-m** *mode* option specifies specific modes for the device group. Valid modes are: **noread**, **cyl**, **trk**, **dif**, **inc**, and **grp[GID]**.

output

The **createpair** command outputs the:

- Data center
- Host name of the host that executed the command
- RM instance used for the command
- Mirror type
- Mirror level
- Command used to execute the remote command
- Output of the remote command

sample output: createpair

This example shows the output of the **createpair** command. The output gives the host and instance that executed the command and the command status.

```
# createpair -g vg_fs5 -h alpha109
Data Center  : DC_two
Host         : alpha109
Instance     : 0
Command      : /usr/bin/paircreate -g vg_fs2 -f never -vl -c 5
createpair completed successful
```

return values

createpair return values

condition	value
Normal termination	0: Pair is created
Abnormal termination	212: Unmatched volume size for pairing 215: No CT groups left for Open Vol use 217: Not enough CT groups in the Raid 222: Invalid volume status 228: Invalid pair status 229: Inconsistent status in group 236: Unmatched volume status in group

deleting the pair relationship — deletepair

The **deletepair** command executes the RM **pairsplit** command for the given pair. The **pairsplit** command:

- Deletes the relationship between two devices
- Returns the devices to SMPL (simplex) status

```
deletepair -g group name [-h host name [-i instance number]] -S|-R-|-P [-v] [-l] [-nomsg]
```

parameters

You can use the following parameters with the **deletepair** command:

createpair parameters

Parameter	Description
-g *group name*	Required. This parameter enables you to specify what device group to execute the command. The group name value must match a device name entry in the configuration file.
[-h *host name***]** **[-i** *instance number***]**	Optional. Executes the command on a specific host and instance. If you do not specify the **-i** *instance number* parameter with the host name parameter, then RM uses the instance numbers specified in the configuration file. If you supply both host name and instance parameters on the command line, then RM uses these values for the remote command execution. In this case, it is not mandatory to define this host name and instance number in the configuration file.
-S|-R|-P	Required. You must provide one of the **-S**, **-R** or **-P** options on the command line to ensure the deletion of the pair. If the **pairsplit** command is executed without these options, it will only suspend the pairs and not delete them. The **-S** option will attempt to delete the entire pair if there is access to both sides. The **-R** and **-P** option force a delete on a specific side of the pair.
-l **-nomsg**	Optional. These parameters enable you to manipulate the operation of the actual **pairsplit** command.

output

The **deletepair** command outputs the:

- Data center
- Host name of the host that executed the command
- RM instance used for the command
- Mirror type
- Mirror level
- Command used to execute the remote command
- Output of the remote command

sample output: deletepair

This example shows the output of the **`deletepair`** command. The output gives the host and instance that executed the command and the status of the command.

```
# deletepair -g vg_fs5 -S
Data Center  : DC_two
Host         : alpha109
Instance     : 0
Command      : /usr/bin/pairsplit -g vg_fs2 -S
deletepair completed successful
```

return values

deletepair return values

condition	value
Normal termination	0: Deletepair succeeded
Abnormal termination	250: Invalid Configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally

finding active application host — find_app

The **find_app** command executes user specific scripts on a remote system in order to interact with an application that is replicated. You must supply the necessary scripts and ensure that it exits with the appropriate exit values.

This command performs a user-defined script called **find_app_***app_name* (located in the **/etc/opt/hpmsdt/exec** directory) on each of the systems defined as "app_hosts" in the configuration file. The user-defined script performs local host commands to determine if the application is running. If the application is active on that system, the script completes with an exit value of 0; if the application is not active on that system, the script aborts with an exit value of 1.

Use this command to determine where the application is currently running. You can then use this information to execute the pre and post split commands on the appropriate host.

You can also execute this command to find the entry point of data into the system.

note The command exits when it finds the first host that returns a 0 value. Because the application can only run on one system at a time, there should only be one host that has a 0 return value.

```
find_app -a application_name [-v] [-o "options"]
```

parameters

You can use the following parameters with the **find_app** command:

Find_app parameters

Parameter	Description
-a *application name*	Required. You must provide this parameter, and the *app name* must match an application name entry in the configuration file.
[-v]	Optional. The **[-v]** verbose parameter displays all attempts to execute on each host. You can use this parameter to debug problems. Without the **[-v]** parameter, the command only returns the successful command text. If the application fails to connect or execute the script on any of the hosts in the list the command will exit with a 255-error code.
[-o *"options"***]**	Optional. This parameter enables you to pass user defined parameters to the script. For security reasons the command will not allow any "**;**" characters in the parameter list.

output

The **find_app** command outputs:

- Without **-v** parameter — the host name of the host for which the script returned a value of 0
- With **-v** parameter — return values for all hosts in the configuration

Sample output: unable to find application

This example shows the output of a **find_app** command that did not find the application running on any host. Since the script did not return a value of 0 from any of the hosts queried, the return value for the command is 1.

```
:>find_app -a myapp1
Application: myapp1 not running on any host
```

Sample output: application found

This example shows the output of a **find_app** command that includes the **-v** parameter. The output gives:

- All host names defined in the configuration file
- The return value from the script for each host

Since the script returned a value of 0 from one of the hosts queried, the return value for the command is 0.

```
:>find_app -a myapp1 -v
Application: myapp1 not running on host: alpha108 return code = 1
Application: myapp1 running on host: alpha109 return code = 0
Application: myapp1 not running on host: alpha155 return code = 1
Application: myapp1 not running on host: alpha156 return code = 1
Application: myapp1 not running on host: alpha154 return code = 1
```

Sample output: application found

This example shows the output of a **find_app** command that includes the **-v** parameter. The output gives:

- All host names defined in the configuration file
- The return value from the script for each host

Since the script could not run on one of the hosts queried, the return value for the command is 255.

note Although the script confirmed that the application is running on one host, it is very risky to assume that alpha155 is not accessing application data as well.

```
:>find_app -a myapp1 -v
Application: myapp1 not running on host: alpha108 return code = 1
Application: myapp1 running on host: alpha109 return code = 0
Application: myapp1 script failed execution on host: alpha155 return code = 255
Application: myapp1 not running on host: alpha156 return code = 1
Application: myapp1 not running on host: alpha154 return code = 1
```

return values

find_app return values (without –v parameter)

condition	value
Normal termination	0: Indicates a host with active application was found 1: Indicates no host with active application was found
Abnormal termination	250: Invalid Configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

find_app return values (with –v parameter)

condition	value
Normal termination	0: All hosts executed the script and at least one returned a 0 1: All hosts executed the script and all returned a 1
Abnormal termination	250: Invalid Configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

preparing an application for a BC suspend — pre_exec

The **pre_exec** command runs a user-defined script called **pre_exec**_*app_name* (located in the **/etc/opt/hpmsdt/exec** directory) on a system defined by the **-h** *host name* parameter. The user-defined script performs local host commands and application interaction to prepare an application for a BC suspend. This could include:

- Stopping the application
- Suspending I/O operations for the application
- Flushing local host file buffers to ensure that all changed data is written to physical disks

note The command returns a 0 value if the application is ready for the split and returns a 1 (or any other user defined return value) if the application is not ready.

You can execute this command to:

- Prepare the application for a BC suspend
- Ensure consistency on the target copy

```
pre_exec -a application_name -h host name [-o "options"]
```

parameters

You can use the following parameters with the **pre_exec** command:

pre_exec parameters

Parameter	Description
-a *app name*	Required. You must provide this parameter, and the *app name* must match an application name entry in the configuration file.
-h *host name*	Required. You can find the host name in the output of the **find_app** command.
[-o *"options"***]**	Optional. This parameter enables you to pass user defined parameters to the script. For security reasons the command will not allow any "**;**" characters in the parameter list.

output

The **pre_exec** command outputs:

- The host name
- The application name
- The command used to execute the **pre_exec**_*app_name* script
- All output from the **pre_exec**_*app_name* script

sample output: pre_exec

This example shows the output of a **pre_exec** command running on host alpha108 with an optional **-shutdown all** option. This is the actual output of the remote script execution.

```
# pre_exec -a fs5 -h alpha108 –o "-shutdown all"
Host        : alpha108
Application : fs5
Command     : /etc/opt/hpmsdt/exec/pre_exec_fs5 –shutdown all
connect to database …….ok
shutdown database …….ok
flush buffers……ok
done
```

return values

pre_exec return values

condition	value
Normal termination	0: Pre_exec script was successful 1: Pre_exec script failed
Abnormal termination	250: Invalid Configuration file 251: Config file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

returning application status to normal after BC suspend — post_exec

The **post_exec** command runs a user-defined script called **post_exec**_*app_name* (located in the **/etc/opt/hpmsdt/exec** directory on a system defined with the **-h** *host name* parameter. The script performs local host commands and application interaction to return an application to normal operation after a BC suspend. This could include:

- Starting an application
- Enabling the I/O operation for an application

note The command returns a 0 value if the application is returned to normal operation and returns a 1 value if the script fails.

Use this command to continue normal operation after a BC suspend process.

```
post_exec -a application_name -h host name [-o "options"]
```

parameters

You can use the following parameters with the **post_exec** command:

post_exec parameters

Parameter	Description
-a *app name*	Required. You must provide this parameter, and the *app name* must match an application name entry in the configuration file.
-h *host name*	Required. You can find the host name in the output of the **find_app** command.
[-o *"options"***]**	Optional. This parameter enables you to pass user defined parameters to the script. For security reasons the command will not allow any "*;*" characters in the parameter list.

output

The **post_exec** command outputs:

- The host name
- The application name
- The command used to execute the **post_exec**_*app_name* script
- All output from the **post_exec**_*app_name* script

sample output: post_exec

This example shows the output of the **post_exec** command running on host alpha108 with an optional **-start** parameter. This is the actual output of the remote script execution.

```
# post_exec -a fs5 -h alpha108 -o "-start"
Host        : alpha108
Application : fs5
Command     : /etc/opt/hpmsdt/exec/post_exec_fs5 -start
connect to database …….ok
start database ……..ok
done
```

return values

post_exec command return values

condition	value
Normal termination	0: Post_exec script was successful 1: Post_exec script failed.
Abnormal termination	251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

verifying the configuration file — msdtverify

The **msdtverify** command verifies the data in the Tools configuration file **/etc/opt/hpmsdt/msdt.cfg** (default), or the file specified on the command line. The command performs the same verification as the **dist_conf** command without distributing the configuration file.

msdtverify [*configuration_file*]

validation checks

The **msdtverify** command performs the following consistency checks upon execution:

msdtverify consistency checks

location	consistency check
Within the entire file	The Application Name is unique for each application in the configuration file, and is only defined once in the file. The Device Group name is unique within the entire file and is defined in only one place.
Within the application section	Data center names used within the group definitions are defined in the data center section. A data center is defined only once in the data center section. The hosts listed in a data center are only defined in that data center and do not occur in other data centers for the same application. Only one device group with a specific mirror level is defined. The application can only have one device group with CA_1 level, one with BC_1 level, one with CA_2 level, and one with BC_2 level.
Within the device group section	Array type should be XP. If omitted, the default parameter is XP. Mirror type is CA or BC. Mirror level is optional. If defined, the value must be CA_1, BC_1, CA_2, or BC_2. There must be a left or right side defined, preferably both.
Within the left or right section	Data center name matches the data center. One or more hosts are defined Each host has one or more instances defined

distributing the configuration file — dist_conf

The **dist_conf** command:

- Validates the data in the Tools configuration file according to the same rules as the **msdtverify** command
- Copies the file to all the hosts defined in the configuration file

Execute this command to keep configuration information consistent between all hosts in the configuration.

```
dist_conf [configuration file]
```

output

The **dist_conf** command outputs a list of hosts that have either successfully received the new configuration file or failed to receive the configuration file.

note You must check any failed hosts manually and re-run the **dist_conf** command.

sample output: command failure

This example shows the output of a **dist_conf** command when one of the hosts fails to receive the file. Since one of the hosts was not available to receive the configuration file, the return value of the command is 1. In this case, the user will have to check that host and update it with the latest files when the host is available again.

```
# dist_conf
Check config file ........OK.
Do you want to copy file /etc/opt/hpmsdt/conf/msdt.cfg to all hosts [yes/no]:yes
alpha108..................ok.
alpha109..................ok.
alpha155..................fail.
alpha156..................ok.
alpha154..................ok.
```

return values

dist_conf return values

condition	Value
Normal termination	0: All systems received the file(s)
Abnormal termination:	250: Invalid configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

distributing a log message — dist_log_msg

The **dist_log_msg** command distributes a message to hosts specified in the configuration file for the application. All hosts that are available at the time of writing will:

- Receive the message
- Write it into an application-specific log file:
  /var/opt/hpmsdt/log/msdt_*application_name***.log**

note You can use this log file to trace events that affected the application devices. It is up to you to determine what messages go into this log file and when to use it.

```
dist_log_msg -a application_name -o "message"
```

parameters

You can use the following parameters with the **dist_log_msg** command:

dist_log_msg parameters

Parameter	Description
-a *app name*	Required. You must provide this parameter, and the *app name* must match an application name entry in the configuration file.
-o *"text message"*	This parameter enables you to specify the desired log message.

output

The **dist_log_msg** command outputs a list of hosts that have either successfully received the new message or failed to receive the message.

note Any host that is not available at the point of writing the message will not be able to record it.

sample output:

This example shows the output of a **dist_log_msg** command when host alpha109 cannot receive the message. Since one of the hosts failed, the return value is 255.

```
:>dist_log_msg -a myapp1 -o "Pair cycle process begin"
Alpha108…….ok
Alpha109…….fail
Alpha155 ……ok
Alpha156…….ok
Alpha190…….ok
Done
```

return values

dist_log_msg return values

condition	value
Normal termination	0: All systems received the message
Abnormal termination:	250: Invalid configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

showing the tools version number — msdtversion

The **msdtversion** command identifies the version number of the Tools installed on a host.

```
msdtversion
```

output

The **msdtversion** command outputs the version number of the Tools installed on a particular host.

sample output

This example shows the output of the **msdtversion** command:

```
:>msdtversion
HP MSDT Tool Kit Version 1.00.00
```

parse the configuration file — msdtinfo

The **msdtinfo** command parses the Tools configuration file and displays specific values. This is useful for scripting a solution that needs to use values defined in the MSDT configuration file.

msdtinfo [-f *configuration file***]** *arguments*

arguments

Use the arguments below to get a specific value:

arguments

argument	description
-A	Get application names in configuration file
-AD	Get application names and their device groups
-AH <*application name*> <*datacenter*>	Get application hosts
-All <*application name*>	Print out all parameters of application
-AType <*application name*> <*device group*>	Get array type
-DA *<device group>*	Get application name for given device group
-DC <*application name*> <*device group*> <*right/left*>	Get data center names
-DG <*application name*>	Get device groups
-DCH <*application name*> <*device group*> <*right/left*>	Get data center host names
-MType <*application name*> <*device group*>	Get mirror type
-RM <*application name*> <*device group*> <*right/left*> <*host name*>	Get host RM instances
-h|?	Help

output

The **msdtinfo** command outputs the information specified in the command arguments.

sample output

This example shows the output of the **msdtinfo -A** command. The output lists the five applications defined in the configuration file.

```
:>msdtinfo -A
oracle_1
myapp1
myapp2
fs1
fs2
```

This example shows the output of the **`msdtinfo –DG fs1`** command. The output lists the device group names for application fs1.

```
:>msdtinfo –DG fs1
vg_fs1
bc1_vg_fs1
ca2_vg_fs1
bc2_vg_fs1
```

This example shows the output of the **`msdtinfo -AD`** command. The output lists the five applications and their device group information as defined in the configuration file.

```
:>msdtinfo -AD
Application: "oracle_1":
Device Groups:
"ora_ca1", Array Type=XP, Mirror Level=(CA_1)
"ora_bc1", Array Type=XP, Mirror Level=(BC_1)
"ora_ca2", Array Type=XP, Mirror Level=(CA_2)
"ora_bc2", Array Type=XP, Mirror Level=(BC_2)
Application: "myapp1":
Device Groups:
"group1", Array Type=XP, Mirror Level=(CA_1)
"group2", Array Type=XP, Mirror Level=(BC_1)
"group3", Array Type=XP, Mirror Level=(CA_2)
"group4", Array Type=XP, Mirror Level=(BC_2)
.....
```

This example shows the output of the **`msdtinfo –DG fs1`** command. The command lists the device group names for application fs1.

```
:>msdtinfo –DG fs1
vg_fs1
bc1_vg_fs1
ca2_vg_fs1
bc2_vg_fs1
```

This example shows the output of the **`msdtinfo –All fs2`** command. The output lists all of the configuration values for the fs2 application. The output is in the format of environment variables that can be easily used in a script.

```
:>msdtinfo –All fs2
App_name=  "fs2"
DC_name[0]="DC_one"
DC_hosts[0]="alpha108,alpha109"
DC_name[1]="DC_two"
DC_hosts[1]="alpha155,alpha156"
DC_name[2]="DC_three"
DC_hosts[2]="alpha154"
DG_level[0]="CA_1"
DG_name[0]="vg_fs2"
DG_type[0]="CA"
DG_left_dc[0]="DC_one"
DG_left_hosts[0]="alpha108:0,alpha109:0"
DG_right_dc[0]="DC_two"
DG_right_hosts[0]="alpha155:1,alpha156:1"
DG_level[1]="BC_1"
DG_name[1]="bc1_vg_fs2"
DG_type[1]="BC"
DG_left_dc[1]="DC_two"
DG_left_hosts[1]="alpha155:2,alpha156:2"
DG_right_dc[1]="DC_two"
DG_right_hosts[1]="alpha155:3,alpha156:3"
DG_level[2]="CA_2"
DG_name[2]="ca2_vg_fs2"
DG_type[2]="CA"
DG_left_dc[2]="DC_two"
DG_left_hosts[2]="alpha155:4,alpha156:4"
DG_right_dc[2]="DC_three"
DG_right_hosts[2]="alpha154:5"
DG_level[3]="BC_2"
DG_name[3]="bc2_vg_fs2"
DG_type[3]="BC"
DG_left_dc[3]="DC_three"
DG_left_hosts[3]="alpha154:6"
DG_right_dc[3]="DC_three"
DG_right_hosts[3]="alpha154:7"
#done.
```

starting, stopping, and restarting — msdtstart, msdtstop, msdtrestart

Use these commands to start, stop, and restart the MSDT server process.

```
msdtstart
msdtstop
msdtrestart
```

return values

msdtstart, msdtstop, and msdt restart return values

condition	value
Normal termination	0: Operation was successful
Abnormal termination:	250: Invalid configuration file 251: Configuration file not found 253: Cannot connect to manager 254: Invalid or insufficient arguments 255: Command failed (partially or totally)

running the MSDT command shell — msdtcmd

The **msdtcmd** command invokes the MSDT interactive command shell. The MSDT command shell enables execution of Tools commands inside a single client session. This reduces the overhead of loading the Java Virtual Machine (JVM) and setting up of client/server connections. Within the MSDT command shell, you can invoke all Tools commands except **dist_conf**. You can also invoke these additional commands:

- **cmdstatus** — Displays the status of the last command executed.
- **cwd** — Displays the current working directory.
- **help** — Displays the help message.
- **redo** — Re-runs the last command executed.

msdtcmd

output

The **msdtcmd** command displays the MSDT prompt from which you can enter the desired Tools command. Tools commands run from the command shell produce that same output as Tools commands invoked from the system shell.

5 Scripts

A script is a combination of instructions and business logic that you can use to automate BC and CA operations in a complex environment. The host reads the script and executes each command as if it was entered manually. It then evaluates the output and return values of the command and makes a logical decision on what to do next.

The goal of developing scripts is to simplify tasks that must be performed on a regular basis, and to ensure that consistent actions are performed in the solution. Once you have created and tested a script, you can use a command scheduler to execute the script at a regular interval. The scheduler will inform you of any failures in the script.

The sample scripts provided with the MSDT tools enable you to manage the solution from a single host, and automate regular procedures in the solution.

predefined scripts

You can find template scripts in the **`/opt/hpmsdt/scripts`** directory. You can use these scripts "straight out of the box" — that is, with no customization — but modified environments may require you to customize, or perhaps even redesign, the scripts.

The MSDT Solution includes the following template scripts:

- A script that displays the entire configuration and the status of all device pairs.
- A script that cycles data through the environment by:
  - Creating the point-in-time copy of data on the BC_1 device
  - Executing a resume-wait-split on the CA_2 link
  - Executing a resume-wait-split on the BC_2 link (if it exists)

functions

To simplify the scripts, some common tasks are grouped into separate functions that you can call from the main part of the script. Functions perform the actions in a script, and the main section of the script applies the logic to the results of the functions. When creating new scripts, you will use some of the standard functions and modify the business logic around the scripts to accomplish new tasks in the solution.

You can find all of the functions for the template scripts in the **`/opt/hpmsdt/scripts/functions`** directory.

caution	Do not modify the original scripts and functions. HP recommends that you create copies of the scripts and functions, and then modify those copies. This will ensure that you can apply future updates without losing any modifications.

variables

Functions use the following global variables to evaluate or set their values:

global variables used in scripts and functions

variable	description
`success=0`	Indicates if a function action has succeeded or failed.
`fail=0`	Indicates additional failure codes not covered by the success variable.
`correct_status=0`	Indicates if the device groups are in the correct status. The VerifyStatus function sets this variable, and most other functions will only operate if this function is set to 0.

You can set the following variables in the configuration file in the user settings for the application.

note Options you enter in the CLI always take precedence over variables in the configuration file.

configuration file variables used in scripts and functions

variable	description
`get_all_status=1`	Indicates if a function action has succeeded or failed.
`verbose=0`	Use this option to determine if you print the diagram every time, or only in the beginning.
`debug=0`	Use this option to print the debug information.
`force=0`	Use this option to force work if you positively identify the application host cannot find the host.

The following variables are default settings for the configuration file:

default variables used in the configuration file

variable	description
`cycle_type=0`	Use this for a suspend all.
`cycle_type=1`	BC_1 is paired.
`msdt_type =x`	Use this variable to indicate how many links are in the configuration. x=the number of links (e.g. 4 = 4 links, 3 = 3 links).
`pre_options="`*`options`*`"`	Pre_exec script options.
`post_options="`*`options`*`"`	Post_exec script options.
`wait_time=x`	Time the eventwaitpair waits for a status change.

displaying configuration settings — disp_conf

One of the most common tasks you will perform is finding the status of an application within the configuration. Since the application exists of a number of hosts that can run the application or control mirror device groups, the use of the Tools scripts is highly recommended. During the install process, the application and all of the device groups used in the application were listed in the Tools configuration file.

The **`disp_conf`** script uses the application name and information in the configuration file to provide a visual output of the current status of the application. It is essential to find the current status of each of the four device groups, and the current location of the application.

The location of the application is the point of data entry to the configuration. From that point on there should be a logical flow of information through the system.

For example:

- If the point of data entry is on site 1, then the CA_1 pair will have a P-vol on site 1.
- If the BC_1 device group is not on the same site then you must ensure that the CA_1 link to site 2 is operational and in PAIR status.
- If the BC_1 pair is on the same data center as the CA_1 P-vol, the status of the CA_1 link is of less importance to the flow of data through the system. (It is, however, very important to the cluster environment and the protection of data.)

output

This is the output of a normal system with **`msdt_type=4`** (indicating you have four links to manage) and **`cycle_type=0`** (indicating you want to suspend all devices for the cycle process to be able to run).

The application is running on Alpha109 in Data center DC_one — therefore the P-vol is in DC_one. Because the BC_1 P-vol is in DC_two, the status of the CA_1 link is very important for data flow to DC_three.

```
       DC_one                        DC_two                       DC_three
====================           ====================          ====================
      alpha108                       alpha155                      alpha154
      alpha109                       alpha156
====================           ====================          ====================
Application fs3 status : Running
      alpha109
====================           ====================          ====================
                      CA_1
 PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                     vg_fs3           PVOL_PSUS                      SVOL_PSUS

                                         |                             |

                                         | BC_1                        | BC_2
                                       PSUS                          PSUS
                                         | bc1_vg_fs3                  | bc2_vg_fs3

                                         |                             |

                                       SVOL_PSUS          CA_2         PVOL_PSUS
                                       PVOL_PSUS    - - - - PSUS  - - - -  SVOL_PSUS
                                                          ca2_vg_fs3
```

cycling device pairs — cycle_pair

The cycling of data through the system ensures that new data is propagated to Site 3 at regular intervals. You can set the data cycling interval to run every few hours, daily, or in a continuous cycle.

Under normal conditions, you will use a scheduler to run the process. On completion, it will exit with:

- A 0 indicating success in cycling the pairs
- A 1 indicating failure to cycle the pairs

In the event of an error, the administrator must be notified, and the problem must be determined and fixed. The administrator uses the output of the **`cycle_pair`** script to find the specific detail of the error encountered. The administrator then uses the Tools commands or RM commands to correct the problem.

Since the **`cycle_pair`** script is designed to start with a given status and end with the same status, it should keep operating without problems unless human interference changes the status of a device group, or a system failure causes an error. The failure of one **`cycle_pair`** command can cause all subsequent cycle commands to fail.

The **`cycle_pair`** command uses parameters from the configuration file to determine the number of device pairs in the configuration and what type of data cycling to perform. **`msdt_type=3`** indicates three device groups (no BC_2 device group). **`msdt_type=4`** indicates four device groups. **`cycle_type=0`** indicates all device pairs except CA_1 must be in suspend status when the cycle process begins. **`cycle_type=1`** indicates that BC_1 must be in pair status when the cycle process begins, while CA_2 and BC_2 must be in suspend status.

If **`cycle_type=0`**, then the first step in the process is to resume the BC_1 device group and wait for it to reach pair status. This operation could take some time, depending on the amount of data that has changed since the last cycle process. When the device pair reaches pair status, you can create the point-in-time copy. The time at which the point-in-time copy is made depends on the resync time, and can potentially be any time after the **`cycle_pair`** script starts. Since the point-in-time process normally requires some application intervention and potential impact to application performance, you should keep this time difference in mind when scheduling the **`cycle_pair`** script. This also may affect the accuracy of the data on the point-in-time copy, since it is not predetermined on what exact timeframe the point-in-time copy is taken.

With **`cycle_type=1`**, the cycle process expects the BC_1 device to already be in pair status and, therefore, can immediately start the critical task of creating the point-in-time copy of the data. Some time is required to collect all of the device group status information and to perform the **`pre_exec`** application tasks.

output

The following is an example of the output of the **`cycle_pair`** script:

```
# ./Cycle_pair -v fs1
================================================================
HP MSDT Configuration (Pair) Cycle
Copyright 2003 Hewlett-Packard Development Company, L.P.
================================================================
System Name:   alpha108
Current Date:  Wed Apr 30 16:54:42 PDT 2003
Arguments:     "-v fs1                          "
Application Name: "fs1"
Creating process lock file....Done
Obtaining status of all device groups for application "fs1"..
   Getting the current status of:[vg_fs1         ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
```

```
   Getting the current status of:[bc1_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
   Getting the current status of:[ca2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
   Getting the current status of:[bc2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS

Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                        DC_three
     ====================           ====================           ====================
           alpha108                        alpha155                       alpha154
           alpha109                        alpha156
     ====================           ====================           ====================
     "fs1" status : Running
           alpha108
     ====================           ====================           ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                        SVOL_PSUS

                                             |                               |

                                             | BC_1                          | BC_2
                                           PSUS                            PSUS
                                             | bc1_vg_fs1                    | bc2_vg_fs1

                                             |                               |

                                         SVOL_PSUS            CA_2           PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
============================================================
Starting the configuration cycling process
============================================================
============================================================
PHASE 1 Make the point-in-time copy of the BC_1 devices
============================================================
  Verifying_status - Required status is....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
                   - Current status is.....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
  CA_1 Pvol is NOT in the same Data center as BC_1 Pvol we DO need check CA_1 status
  Verifying_status - Successful
++++++++++++++++++++++++++++++++
Resume bc1_vg_fs1 ....Done
++++++++++++++++++++++++++++++++
--------------------------------
Collecting status for BC_1 pair
```

```
--------------------------------
   Getting the current status of:[bc1_vg_fs1     ]...PVOL_COPY...SVOL_COPY...Link=COPY

Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_COPY                          SVOL_PSUS

                                            V                                   |
                                            V
                                            V BC_1                              | BC_2
                                          COPY                                PSUS
                                            V bc1_vg_fs1                        | bc2_vg_fs1
                                            V
                                            V                                   |
                                            V
                                         SVOL_COPY            CA_2            PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                                 ca2_vg_fs1
++++++++++++++++++++++++++++++++++++++++++++++++
Waiting for bc1_vg_fs1 to reach PAIR status .....Done
++++++++++++++++++++++++++++++++++++++++++++++++
--------------------------------
Collecting status for BC_1 pair
--------------------------------
   Getting the current status of:[bc1_vg_fs1     ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PAIR                          SVOL_PSUS
                                             |
                                             |                                  |
                                             |
                                             | BC_1                             | BC_2
                                           PAIR                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1
                                             |
                                             |                                  |
                                             V
                                         SVOL_PAIR            CA_2            PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                                 ca2_vg_fs1
============================================================
Collecting data for ALL device groups for application fs1
This is the critical point of suspend the BC_1 devices
It is best to force update on all device groups
============================================================
   Getting the current status of:[vg_fs1         ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
   Getting the current status of:[bc1_vg_fs1     ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
   Getting the current status of:[ca2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
   Getting the current status of:[bc2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PAIR                          SVOL_PSUS
                                             |
                                             |                                  |
                                             |
                                             | BC_1                             | BC_2
                                           PAIR                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1
                                             |
                                             |                                  |
                                             V
                                         SVOL_PAIR            CA_2            PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
  ------------------------------------------------------------------------
  Preparing the application for the BC_1 split
  Step 1 : Find the host that is running the application
  Application fs1 Found running on alpha108 in data center DC_one
  ------------------------------------------------------------------------
  Step 2 : Run pre_exec_fs1 on alpha108
Host        : alpha108
Application : fs1
Command     : /etc/opt/hpmsdt/exec/pre_exec_fs1 -a all -shutdown
This is the output from the pre_exec script
Shutting down the application... Done
Write date file... Done
Sync devices... Done
Done
  pre_exec scripts worked -- Continue to the next step
  ------------------------------------------------------
  Verifying_status - Required status is....CA_1= PAIR   BC_1= PAIR   CA_2= PSUS   BC2= PSUS
                   - Current status is.....CA_1= PAIR   BC_1= PAIR   CA_2= PSUS   BC2= PSUS
  CA_1 Pvol is NOT in the same Data center as BC_1 Pvol we DO need check CA_1 status
```

```
  Verifying_status - Successful
++++++++++++++++++++++++++++++++++++++++++++++++
  Suspending bc1_vg_fs1 ......Done
++++++++++++++++++++++++++++++++++++++++++++++++
--------------------------------
Collecting status for  BC_1 pair
--------------------------------
   Getting the current status of:[bc1_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS

Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                         DC_three
     ====================           ====================           ====================
           alpha108                        alpha155                       alpha154
           alpha109                        alpha156
     ====================           ====================           ====================
     "fs1" status : Running
           alpha108
     ====================           ====================           ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_PSUS

                                             |                                  |

                                             | BC_1                             | BC_2
                                           PSUS                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1

                                             |                                  |

                                         SVOL_PSUS            CA_2            PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
******************************************************
BC_1 group is Suspended ----- Phase 1 of Cycle completed
******************************************************
  ------------------------------------------------------------------
  Recovering application after BC_1 split
  Step 1 : Find the host that is running the application
  Application fs1 Found running on alpha108 in data center DC_one
  ------------------------------------------------------------------
  Step 2 : Run post_exec_fs1 on alpha108
Host        : alpha108
```

```
Application : fs1
Command     : /etc/opt/hpmsdt/exec/post_exec_fs1 -start
This is the output of the post_exec script
Start application ... Done
Check appication ... Done
Done
  post_exec scripts worked -- Continue to the next step
  ------------------------------------------------------
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Post_exec successful after suspend
++++++++++++++++++++++++++++++++++++++++++++++++++++++++
============================
PHASE 2 Cycle the CA_2 link
============================
  Verifying_status - Required status is....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
                   - Current status is.....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
  CA_1 Pvol is NOT in the same Data center as BC_1 Pvol we DO need check CA_1 status
  Verifying_status - Successful
++++++++++++++++++++++++++++++++
Resume ca2_vg_fs1 ....Done
++++++++++++++++++++++++++++++++
----------------------------------
Collecting status for  device pair
----------------------------------
   Getting the current status of:[ca2_vg_fs1     ]...PVOL_COPY...SVOL_COPY...Link=COPY
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                           CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                          vg_fs1           PVOL_PSUS                         SVOL_PSUS

                                             |                                 |

                                             | BC_1                            | BC_2
                                           PSUS                              PSUS
                                             | bc1_vg_fs1                      | bc2_vg_fs1

                                             |                                 |

                                          SVOL_PSUS            CA_2           PVOL_PSUS
                                          PVOL_COPY    >>>>>>> COPY  >>>>>>> > SVOL_COPY
                                                                 ca2_vg_fs1
  ++++++++++++++++++++++++++++++++++++++++++++++++
  Waiting for ca2_vg_fs1 to reach PAIR status ....Done
  ++++++++++++++++++++++++++++++++++++++++++++++++
  ---------------------------------
  Collecting status for  device pair
  ---------------------------------
   Getting the current status of:[ca2_vg_fs1     ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_PSUS

                                             |                                  |

                                             | BC_1                             | BC_2
                                           PSUS                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1

                                             |                                  |

                                         SVOL_PSUS            CA_2           PVOL_PSUS
                                         PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                                                              ca2_vg_fs1
  ++++++++++++++++++++++++++++++++++++++++++++++++
  Suspending ca2_vg_fs1 ......Done
  ++++++++++++++++++++++++++++++++++++++++++++++++
  ---------------------------------
  Collecting status for  device pair
  ---------------------------------
   Getting the current status of:[ca2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_PSUS

                                             |                                  |

                                             | BC_1                             | BC_2
                                           PSUS                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1

                                             |                                  |

                                         SVOL_PSUS            CA_2           PVOL_PSUS
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
******************************************************
CA_2 group is Suspended ----- Phase 2 of Cycle completed
******************************************************
============================
PHASE 3 Cycle the BC_2 link
============================
  Verifying_status - Required status is....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
                   - Current status is.....CA_1= PAIR   BC_1= PSUS   CA_2= PSUS   BC2= PSUS
  CA_1 Pvol is NOT in the same Data center as BC_1 Pvol we DO need check CA_1 status
  Verifying_status - Successful
++++++++++++++++++++++++++++++++
Resume bc2_vg_fs1 ....Done
++++++++++++++++++++++++++++++++
----------------------------------
Collecting status for device pair
----------------------------------
   Getting the current status of:[bc2_vg_fs1     ]...PVOL_COPY...SVOL_COPY...Link=COPY
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_COPY
                                                                               ^
                                             |                                 ^
                                                                               ^
                                             | BC_1                            ^ BC_2
                                           PSUS                              COPY
                                             | bc1_vg_fs1                      ^ bc2_vg_fs1
                                                                               ^
                                             |                                 ^

                                         SVOL_PSUS            CA_2            PVOL_COPY
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
++++++++++++++++++++++++++++++++++++++++++++++++++
Waiting for bc2_vg_fs1 to reach PAIR status ....Done
++++++++++++++++++++++++++++++++++++++++++++++++++
----------------------------------
Collecting status for device pair
----------------------------------
   Getting the current status of:[bc2_vg_fs1     ]...PVOL_PAIR...SVOL_PAIR...Link=PAIR
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_PAIR
                                                                                 ^
                                             |                                   |
                                                                                 |
                                             | BC_1                              | BC_2
                                           PSUS                                PAIR
                                             | bc1_vg_fs1                        | bc2_vg_fs1
                                                                                 |
                                             |                                   |
                                                                                 |
                                         SVOL_PSUS            CA_2           PVOL_PAIR
                                         PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                              ca2_vg_fs1
++++++++++++++++++++++++++++++++++++++++++++++++++
Suspending bc2_vg_fs1 ......Done
++++++++++++++++++++++++++++++++++++++++++++++++++
----------------------------------
Collecting status for device pair
----------------------------------
   Getting the current status of:[bc2_vg_fs1     ]...PVOL_PSUS...SVOL_PSUS...Link=PSUS
```

```
Application Status:
"fs1" is running on alpha108 in data center DC_one.
            DC_one                          DC_two                          DC_three
     ====================            ====================            ====================
           alpha108                        alpha155                        alpha154
           alpha109                        alpha156
     ====================            ====================            ====================
     "fs1" status : Running
           alpha108
     ====================            ====================            ====================
                          CA_1
      PVOL_PAIR  - ------- PAIR  ------- > SVOL_PAIR
                         vg_fs1           PVOL_PSUS                          SVOL_PSUS

                                             |                                  |

                                             | BC_1                             | BC_2
                                           PSUS                               PSUS
                                             | bc1_vg_fs1                       | bc2_vg_fs1

                                             |                                  |

                                          SVOL_PSUS            CA_2            PVOL_PSUS
                                          PVOL_PSUS    - - - - PSUS  - - - -   SVOL_PSUS
                                                                ca2_vg_fs1
*******************************************************
BC_2 group is Suspended ----- Phase 3 of Cycle completed
*******************************************************
*** Complete ***  Wed Apr 30 17:06:28 PDT 2003 Cycle for application fs1 completed sucessfull
#
```