



MetroCluster TieBreaker

NetApp
2.1 – August 15, 2011

TABLE OF CONTENTS

1	DEFINITIONS	4
2	OVERVIEW	4
2.1	CAVEATS	4
3	PREREQUISITES	5
4	INSTALLATION	6
4.1	STEP 1 – INSTALL THE OPERATIONS MANAGER SCRIPT PLUGIN	6
4.2	STEP 2 – RUNNING THE INSTALL SCRIPT	9
5	UNINSTALL	12
6	CONFIGURATION	13
6.1	CONFIGURATION	13
6.2	MONITOR	13
6.3	ENABLED	14
6.4	POLLINGINTERVAL	14
6.5	CFODABORTTIMEOUT	14
6.6	CFODSUCCESSWAITTIMEOUT	14
6.7	TESTMODE	14
6.8	SITE0/SITE1	15
6.9	PINGTIMEOUT	15
6.10	PINGRETRIES	15
6.11	FILER	15
6.12	GATEWAYS	16
6.13	GATEWAY	16
6.14	HOSTNAME	16
6.15	USERNAME	16
6.16	PASSWORD	16
6.17	ENCRYPTEDPASSWORD	17
6.18	SSL	17
6.19	CONNECTTIMEOUT	17
6.20	CONNECTRETRIES	17
6.21	IGNOREDAGGRS	17
6.22	IGNOREDAGGR	18
7	OPERATIONS	19
7.1	EVENTS AND ALERTING	19
8	MONITORING PROCEDURE	20
8.1	SITE REACHABILITY TESTING	20

8.2	FILER STATUS UPDATE	20
8.3	CFOD CHECK	21
8.4	MIRROR DEGRADATION CHECK	21
8.5	SITE FAILOVER	21
9	DFM EVENTS.....	22
9.1	AUTHENTICATION-FAILURE	22
9.2	API-ERROR	22
9.3	COMMUNICATION-RESTORED	22
9.4	CONFIGURATION-FAILED	22
9.5	ISOLATION-DETECTED	22
9.6	ISOLATION-RESOLVED	23
9.7	LINK-DOWN	23
9.8	LINK-UP	23
9.9	MIRROR-DEGRADED	23
9.10	THRESHOLD-EXCEEDED	23
9.11	MIRROR-NORMAL	23
9.12	CFOD-SUCCEEDED	24
9.13	CFOD-FAILED	24
9.14	CFOD-UNKNOWN	24
9.15	CFOD-ABORTED	24
APPENDIX A – CONFIGURATION FILE EXAMPLE		25
APPENDIX B – ONTAP API (ONTAPI) CALLS USED		26

1 DEFINITIONS

- **CFOD** – Cluster Failover on Disaster. This type of takeover forces the surviving controller to take ownership of its partner's disks and begin serving its data. It is used when the surviving controller in a MetroCluster cannot communicate with its partner.
- **Gateway** – An IP router providing access to a Site. Zero or more Gateways can be configured, and are tested for reachability.
- **Monitor** – A configuration specifying a pair of Sites, each containing a Controller and an optional list of Gateways. Multiple Monitors can be defined
- **Monitor Group** – A DFM host group created by MetroCluster TieBreaker. These groups are the source of site and monitor level events.
- **Site** – A location defined within a Monitor containing a Controller and a list of Gateways. A Site includes gateway monitoring parameters.

2 OVERVIEW

The purpose of the MetroCluster Tiebreaker (MCTB) solution is to provide automatic failover of the MetroCluster under conditions when the MetroCluster cannot perform automatic failover itself. This can occur when the interconnect links between the two controllers goes down. Without communications between the two controllers, a surviving controller will not perform a takeover automatically.

MCTB continuously monitors the MetroCluster controllers and corresponding network gateways from a DFM server at a 3rd location. When MCTB detects conditions that required a Cluster Failover on Disaster (CFOD), it issues the necessary ZAPI commands to initiate the CFOD.

MCTB uses a configuration file to determine what to monitor and under what conditions to initiate a CFOD. Log messages and DFM events are generated when necessary to keep the operator informed as to the state of the MetroCluster and MCTB.

MCTB must be installed on a DFM host, and runs in the background (and can run as a Windows Service or Unix/Linux daemon). A DFM script-plugin interface provides control and monitoring of MCTB, and specifies the DFM events that can be sent from MCTB to DFM.

An Operations Manager Group is created for each monitor and its two sites defined in the configuration. The groups have the same names as the name of the monitor and sites as specified in the configuration. Each site group contains the controller defined in the Site. This provides a view of the MetroCluster, its status, and the status of the two sites and controllers.

2.1 CAVEATS

No automated monitoring system can guarantee that automatic failover will always occur when necessary and only when necessary. The MetroCluster TieBreaker monitors specific criteria and, when met, initiates forced takeover. Conditions outside of the specific criteria being monitored, but that could indicate a disaster event, will not result in an automatic takeover. Other conditions may prevent successful automatic takeover. In these cases, human intervention and manual takeover will be necessary.

This solution does not provide any features to assist in performing a giveback after a CFOD. Care must be taken when restoring the MetroCluster in order to avoid split-brain syndrome. Detailed instructions can be found in [TR-3548, Best Practices for MetroCluster Design and Implementation](#).

3 PREREQUISITES

The MetroCluster TieBreaker application is delivered as a DFM Script Plugin zip file, and is supported on DFM version 4.0 and later, installed on Windows Server 2008 R2, or any Unix/Linux operating system supported by DFM.

On Windows systems, Perl version 5.10 or later is also required.

4 INSTALLATION

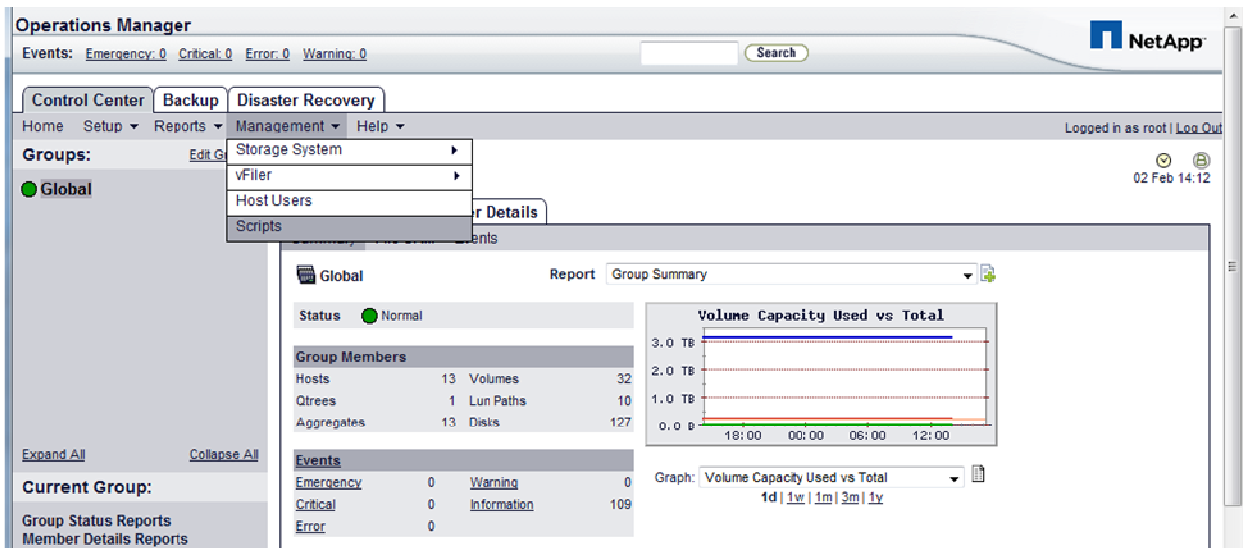
NOTE: On Windows systems, if you are uninstalling the script plugin, or performing an upgrade, you must first stop the MetroCluster TieBreaker Service. You cannot upgrade or uninstall MCTB with the service running. If you are upgrading, you do not need to uninstall the Windows service.

To install the plugin use the following steps:

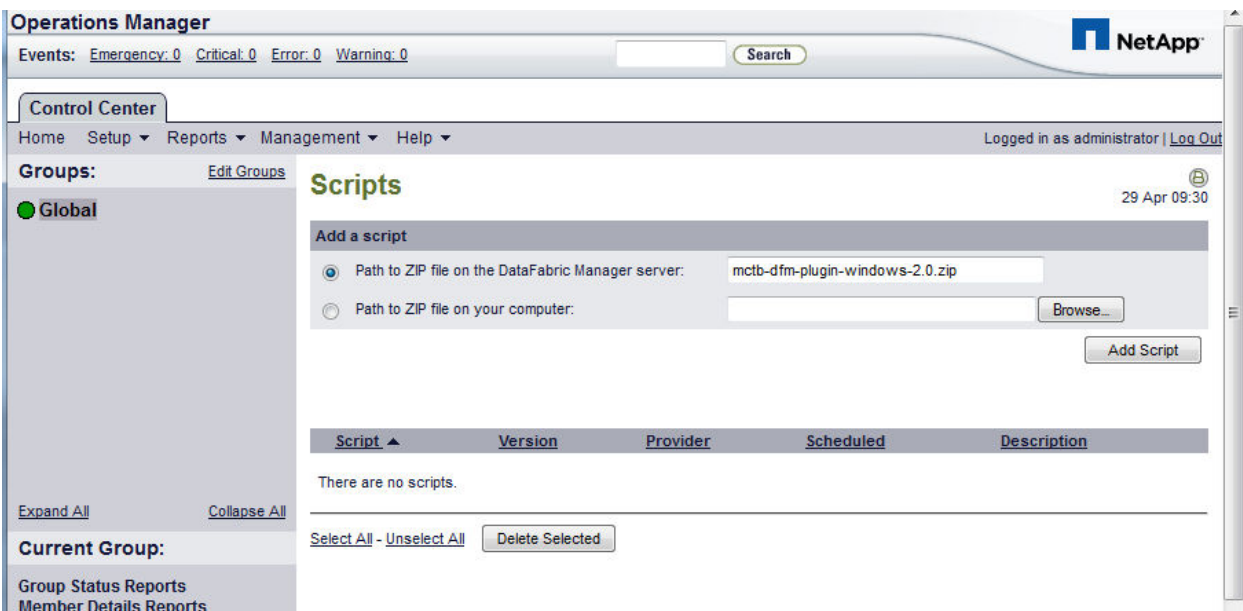
4.1 STEP 1 – INSTALL THE OPERATIONS MANAGER SCRIPT PLUGIN

The MetroCluster TieBreaker is delivered as a DFM Script Plugin and is installed from the Operations Manager user interface.

1. Download the MCTB DFM script plugin zip file to a known location on your computer or the DFM server.
2. Select **Management->Scripts** from the Operations Manager menu:



3. Next, select the appropriate path option (based on where you downloaded the file in step 1), enter the path to the zip file, and select **Add Script**.



4. Next, you will be asked to confirm adding the script. Select **Add**.

The screenshot shows the 'Scripts - Confirm Adding a Script' dialog in the NetApp Operations Manager. The dialog displays the following information for the 'MetroCluster TieBreaker' script:

- Version: 2.0.0
- Provider: NetApp, Inc.
- Description: A server application that monitors server and MetroCluster nodes for connectivity... See help for usage.
- Required Privilege(s): Read/Write/Delete
- Installation Directory: MetroCluster-TieBreaker
- Interpreter: perl
- Output MIME type: text/plain

At the bottom right of the dialog are 'Cancel' and 'Add' buttons. The left sidebar shows the 'Global' group selected under 'Groups'.

5. The final screen shows the installed script. Jobs can now be created using this script to perform various tasks on the MetroCluster TieBreaker.

The screenshot shows the 'Scripts' page in the NetApp Operations Manager. It displays the installed version 2.0.0 of the 'MetroCluster TieBreaker' script (3072). Below this, there is an 'Add a script' section with two radio buttons:

- ☒ Path to ZIP file on the DataFabric Manager server:
- ☐ Path to ZIP file on your computer:

An 'Add Script' button is located at the bottom right of the 'Add a script' section. Below this, a table lists the installed scripts:

Script	Version	Provider	Scheduled	Description
☐ MetroCluster TieBreaker	2.0.0	NetApp, Inc.	No	A server application that monitors server and MetroCluster nodes for connectivity... See help for usage.

The left sidebar shows the 'Global' group selected under 'Groups'.

6. Selecting the underlined script title on the Scripts page will bring you to the Script Details screen.

The screenshot shows the NetApp Operations Manager interface. The top navigation bar includes 'Control Center' and a search box. The left sidebar shows 'Groups' with 'Global' selected. The main content area is titled 'Script - Script Details' and displays the following information for the 'MetroCluster TieBreaker' script:

Version	2.0.0
Provider	NetApp, Inc.
Description	A server application that monitors server and MetroCluster nodes for connectivity... See help for usage.
Schedules	None
Installed On	29 Apr 09:46 (by BAVSPHERESERVERadministrator)
Installation Directory	MetroCluster-TieBreaker
Required Privileges	Read/Write/Delete
Interpreter	perl
Output MIME type	text/plain

Below the script details, there are links for 'Expand All' and 'Collapse All', and a section for 'Current Group' with links for 'Group Status Reports' and 'Member Details Reports'.

7. Clicking on the question mark (?) next to the script title will provide usage information:

The screenshot shows the 'MetroCluster TieBreaker Help' window in Mozilla Firefox. The window title is 'MetroCluster TieBreaker Help - Mozilla Firefox' and the address bar shows the URL 'http://10.61.167.190:8080/dfm/report/script-details/3072?group=0'. The help text is as follows:

MetroCluster TieBreaker (MCTB) Application:

This DFM script plug-in is used to monitor the MetroCluster TieBreaker (MCTB) application. MCTB is a server process that monitors MetroCluster environments for certain conditions and, when met, initiates CFOD (Cluster Failover On Disaster). This script is installed on a DFM server and allows various aspects of MCTB to be controlled and monitored.

This MCTB application will raise DFM events when certain conditions are detected. Not all events are reset automatically when the condition ceases to exist. It is up to the administrator to properly clear these events.

The invocation of this DFM script-plugin is done by creating a DFM "job" which is executed through the DFM interface. The following command line options are supported:

- * start - start the MCTB application.
- * stop - stop the MCTB application.
- * status - print the status of the MCTB application (running or stopped)
- * validate - validate the configuration
- * version - print the version
- * listlogs - list MCTB log files
- * showlog - show the contents of the current log file, or a specific log

Please refer to the README.doc to understand the installation and administration of this application.

This provides quick access to information necessary for setting up jobs using this script.

4.2 STEP 2 – RUNNING THE INSTALL SCRIPT

On both Windows and Unix/Linux systems it is necessary to run an install script after the DFM script plugin has been installed. On Windows systems, this step installs the MCTB Windows Service. On Unix/Linux systems it installs a private JRE, sets file permissions, and installs the daemon. In both cases, the script is run from the Operations Manager script interface using the following steps:

1. Select any group under **Groups** on the left under the **Control Center** tab. Select the **Member Details** tab and the **Scripts** menu item.

The screenshot displays the NetApp Operations Manager web interface. At the top, the 'Operations Manager' header includes event counts (Emergency: 0, Critical: 0, Error: 0, Warning: 0) and a search bar. The 'Control Center' tab is active, showing a left sidebar with 'Groups' (Global) and 'Current Group' (Group Status Reports, Member Details Reports). The main content area is titled 'Jobs, All' and features a 'Group Status' tab. Below this, a 'Scripts' sub-tab is selected, showing a 'Global' group with a 'Start a Job' button. A 'Report' dropdown is set to 'Jobs, All'. A table with columns 'Name', 'Status', 'Start Time', 'Schedule', and 'Group(s)' is shown, containing the message 'There are no jobs.' At the bottom, there are buttons for 'Select All - Unselect All', 'Cancel Selected', 'Re-run Selected', and 'Delete Selected'.

- Click the **Start a Job** button. Enter a **Job Name** (e.g. install), select **MetroCluster TieBreaker** as the **Script to Execute**, and **install** for **Command Line Arguments**. Ignore **Group(s) to Use for Input**. Click the **Start Job** button at the bottom right of the screen.

Operations Manager NetApp

Events: [Emergency: 0](#) [Critical: 0](#) [Error: 0](#) [Warning: 0](#)

Control Center Logged in as administrator | [Log Out](#)

Home Setup Reports Management Help

Groups: [Edit Groups](#)

Global

[Expand All](#) [Collapse All](#)

Current Group:

- Group Status Reports
- Member Details Reports
- Edit Settings
- Edit Membership
- Edit Storage System Configuration
- Edit vFiler Configuration
- Run a Command

Scripts - Start a Job

29 Apr 09:57

Specify the Job Parameters

Job Name:

Script to Execute:

Command Line Arguments:

Group(s) to Use for Input:

☒ Global

☐ Apply to subgroups

The job will run and the status table will appear. The status may say **Running** or **Success**. On Windows, there is no need to wait for the screen to refresh the status from **Running** to **Success**; the job will complete quickly. In Unix/Linux systems, the process may take longer, but should complete well within a minute.

- Select **Success** or **Running** under the **Status** column in the **Script Jobs** table.

Operations Manager NetApp

Events: [Emergency: 0](#) [Critical: 0](#) [Error: 0](#) [Warning: 0](#)

Control Center Logged in as administrator | [Log Out](#)

Home Setup Reports Management Help

Groups: [Edit Groups](#)

Global

[Expand All](#) [Collapse All](#)

Current Group:

- Group Status Reports
- Member Details Reports

Jobs, All

29 Apr 09:58

Script Job 3074 started.

Group Status **Member Details**

Physical Systems Virtual Systems File Systems Aggregates LUNs Scripts

Global

Script Jobs, All: 1-1 of 1

Name	Status	Start Time	Schedule	Group(s)
install	Success	29 Apr 09:58	Manual Execution	Global

Script Jobs, All: 1-1 of 1

[Select All](#) [Unselect All](#)

4. Select the **Output** link next to **Success** on the **Status** line of the **Job Details** screen.

Operations Manager
Events: [Emergency: 0](#) [Critical: 0](#) [Error: 0](#) [Warning: 0](#) [Search](#)

Control Center
Home Setup Reports Management Help Logged in as administrator | [Log Out](#)

Groups: [Edit Groups](#)
● **Global**

Script - Job Details 29 Apr 10:00

Group Status **Member Details**

Physical Systems Virtual Systems File Systems Aggregates LUNs **Scripts**

Details for Script Job 3074

Script	MetroCluster TieBreaker
Status	Success (Output)
Command Line Arguments	install
Input Group(s)	Global
Schedule	Manual Execution
Scheduled/Run by	BAV/SPHERESERVER/administrator
Start Time	29 Apr 09:58

Events Generated by This Job
None

[Expand All](#) [Collapse All](#)

Current Group:
[Group Status Reports](#)
[Member Details Reports](#)

5. The results will show in a browser window (this example is from a Windows installation):

Firefox

<http://10.61.167.190:8080/dfm/report/view/script-job-output/3078?group=0>

```
Installing MCTB Windows Service
=====
Installing the service 'MCTB' ...
The service 'MCTB' has been installed.
```

5 UNINSTALL

To uninstall the MetroCluster TieBreaker, you must first stop and uninstall the daemon or Windows service. This can be done using standard service/daemon interfaces, or from the Operations Manager script interface using the *stop* argument. Once the service/daemon has been stopped, the uninstall is done, also using the script interface, but with the command argument, *uninstall*.

After stopping and uninstalling the daemon or service, you can remove the plugin by selecting the **Management->Scripts** menu item in Operation Manager, selecting the plugin, and clicking **Delete Selected**.

6 CONFIGURATION

The configuration for MetroCluster TieBreaker is done through an XML file named *config.xml* located in the *conf* directory under the installation directory. The file consists of a list of *monitors*, each of which can monitor a single MetroCluster. Each monitor contains the definition for 2 sites and additional monitoring parameters. The *name* attribute of the monitor will correspond to an Operations Manager Group created when the monitor is started.

Within each Site is a Filer and an optional list of gateway routers. Monitoring parameters for the gateways are also defined at this level. Gateways are specified as a list of IP addresses for each gateway router providing access to the site

Each Filer contains parameters for connecting and monitoring.

The plugin script has a validate mode (command line argument *validate*), which will report any problems with the configuration and exit. It is recommended that any changes to the configuration be validated.

The remainder of this chapter describes the configuration file in detail. A complete example of a configuration file can be found in Appendix A.

6.1 CONFIGURATION

Required: true

The top level element in the xml configuration file is **configuration**, and contains a single attribute, **version**, which holds the value 1.0. The configuration element contains a list of **monitors**.

Example:

```
<configuration version="1.0">
  <monitor name="Cluster1">
    . . .
  </monitor>
  <monitor name="Cluster2">
    . . .
  </monitor>
</configuration>
```

6.2 MONITOR

Required: true

The **monitor** element contains a required **name** attribute and contains required **enabled** and **pollingInterval** parameter elements and can contain an optional **testMode** parameter element. The monitor element also contains **site0**, **site1** subelements.

Example:

```
<monitor name="Cluster1">
  <enabled>true</enabled>
  <pollingInterval>10</pollingInterval>
  <testMode>true</testMode>
  <site0 name="Chicago">
    . . .
  </site0>
  <site1 name="Oak Park">
    . . .
  </site1>
</monitor>
```

6.3 ENABLED

Required: true

The **enabled** parameter is a mandatory element defined within a monitor. When set to a value other than **true**, the monitor will not start. In order to start a previously disabled monitor, this value must be set to true and MCTB restarted.

Example:

```
<enabled>true</enabled>
```

6.4 POLLINGINTERVAL

Required: true

The **pollingInterval** parameter sets the polling interval for the monitor, in seconds.

Example

```
<pollingInterval>10</pollingInterval>
```

6.5 CFODABORTTIMEOUT

Required: true

The **cfodAbortTimeout** parameter sets that maximum allowed mirror degradation time before CFOD is disabled. When aggregate mirrors have been degraded for this amount of time (in seconds), automatic CFOD is disabled. This allows a balance between service availability and loss of data (a takeover with degraded mirrors can result in data loss).

Example:

```
<cfodAbortTimeout>90</cfodAbortTimeout>
```

6.6 CFODSUCCESSWAITTIMEOUT

Required: true

The **cfodSuccessWaitTimeout** value is used to determine how long to wait for confirmation after issuing a CFOD, in seconds. If the CFOD completes within this time, the result will be either CFOD succeeded or CFOD failed. If this timeout is reached, the software will stop waiting for a response and the result will be CFOD result unknown.

Example:

```
<cfodSuccessWaitTimeout>120</cfodSuccessWaitTimeout>
```

6.7 TESTMODE

Required: false, **Default:** false

The **testMode** parameter is optional and defaults to false. When set to true, the monitor will log and report events, but will NOT invoke a CFOD. This mode can be used for testing the configuration and confirming accurate takeover attempts, without requiring recovery.

Example:

```
<testMode>true</testMode>
```

6.8 SITE0/SITE1

Required: true

The two monitor subelements, **site0** and **site1**, are required, and contain a single attribute, **name**. Each site contains required a **filer** subelement, and can contain optional **pingTimeout** and **pingRetries** parameter elements, and an optional **gateways** subelement.

Example:

```
<site0 name="Chicago">
  <pingTimeout>5</pingTimeout>
  <pingRetries>3</pingRetries>
  <filer name="chgfiler1">
    . . .
  </filer>
  <gateways>
    . . .
  </gateways>
</site0>
```

6.9 PINGTIMEOUT

Required: false, **Default:** 3 seconds

The **pingTimeout** parameter is optional, and specifies the timeout, in seconds, used when attempting to contact the gateways. This value is only used for gateway testing, and defaults to 3 seconds.

Example

```
<pingTimeout>3</pingTimeout>
```

6.10 PINGRETRIES

Required: false, **Default:** 3 retries (4 attempts total)

The **pingRetries** parameter is optional, and specifies the number of retries that will be made when attempting to contact the gateways. This value is only used for gateway testing, and defaults to 3.

Example

```
<pingTimeout>3</pingTimeout>
```

6.11 FILER

Required: true

The Site subelement **Filer** is required and contains a single attribute, **name**, which must correspond to the actual name of the controller. Each Filer configuration contains parameter elements **hostname**, **username**, **password** or **encryptedPassword** (see descriptions below), and **ssl**, which allow direct contact to the controller. Optional parameters **connectTimeout** and **connectRetries** can be used to control how the Filer is tested for reachability.

Example:

```
<filer name="chgfiler1">
  <hostname>10.10.10.30</hostname>
  <username>root</username>
  <encryptedPassword>DnPw20J-OVkX0UmJb-Q</encryptedPassword>
  <ssl>false</ssl>
  <connectTimeout>10</connectTimeout>
  <connectRetries>1</connectRetries>
</filer>
```

6.12 GATEWAYS

Required: false, **Default:** none

The Site subelement **gateways** starts the list of gateway addresses for the site. If no gateways exist, or gateway monitoring is not desired, then this can be an empty element (e.g. <gateways/>) or can be omitted entirely.

Example:

```
<gateways>
  <gateway>10.10.10.1</gateway>
  <gateway>10.10.10.2</gateway>
</gateways>
```

6.13 GATEWAY

Required: false, **Default:** none

The **gateway** element contains a single IP address or hostname of a gateway router for the site. If *any* of the site's gateways are reachable, the site is considered reachable. IP addresses should be used to avoid issues with host name resolution.

Example:

```
<gateway>10.10.10.1</gateway>
```

6.14 HOSTNAME

Required: true

The **hostname** parameter is required, and specifies the hostname or IP address of the controller. Using an IP address is recommended, as it removes dependencies on host name resolution.

Example:

```
<hostname>10.10.10.31</hostname>
```

6.15 USERNAME

Required: true

The **username** parameter is required and specifies the account that will be used to execute the ONTAP API calls required for monitoring (See Appendix B for a list of API calls this user must be authorized to execute).

Example:

```
<username>monitor</username>
```

6.16 PASSWORD

Required: one of **password** or **encryptedPassword** is required

The **password** parameter is used to set a new password, and will only exist in the file temporarily. When MCTB starts up, it checks for instances of the password element. All password elements found are encrypted and the configuration file is immediately rewritten. Each password element is replaced by an **encryptedPassword** element containing encrypted text. Either password or encryptedPassword is required.

Example:

```
<password>secret</password>
```

6.17 ENCRYPEDPASSWORD

Required: one of **password** or **encryptedPassword** is required

The **encryptedPassword** element contains the encrypted password used to authenticate the user specified in the username element. See password above for details on how passwords become encrypted. Either encryptedPassword or password is required.

Example:

```
<encryptedPassword>DnPw20J</encryptedPassword>
```

6.18 SSL

Required: true

The **ssl** element is required. When set to true, **SSL** will be used to communicate with the filer.

Example:

```
<ssl>true</ssl>
```

6.19 CONNECTTIMEOUT

Required: false, **Default:** 10 sec

This timeout controls how long the controller reachability test will wait for a response. When this timeout is exceeded, the controller is considered not reachable.

Example:

```
<connectTimeout>10</connectTimeout>
```

6.20 CONNECTRETRIES

Required: false, **Default:** 1

The number of times the controller reachability test will attempt to connect. The total number of connections is this value plus 1 (the initial attempt). If the controller does not respond within this number of retries, it is considered not reachable. The total time taken to establish a not reachable condition will be approximately connectTimeout X (connectRetries+1) seconds.

Example:

```
<connectRetries>1</connectRetries>
```

6.21 IGNOREDAGGRS

Required: false, **Default:** none

A list of aggregates that will be ignored with regard to mirror degradation. Not all aggregates are required to be mirrored. Unmirrored aggregates should be added to this list. . If no aggregates are to be ignored, this can be an empty element (e.g. <ignoredAggrs/>) or can be omitted entirely.

Example:

```
<ignoredAggrs>  
  <ignoredAggr>aggr1</ignoredAggr>  
</ignoredAggrs>
```

6.22 IGNOREDAGGR

Required: false, **Default:** none

A single aggregate to ignore.

Example:

```
<ignoredAggr>aggr1</ignoredAggr>
```

7 OPERATIONS

The MetroCluster TieBreaker is designed to be completely automated. The monitoring daemon starts on system boot (if the service or daemon has been configured, see *STEP 2 – Running the Install Script* above) and begins monitoring the configured controllers immediately. If the service cannot start due to a configuration error, a DFM event is issued (see the *Configuration-Failed* event below). Otherwise, the system operates in the background, only issuing DFM events when abnormal conditions arise.

Each time MCTB starts, the configuration is read and enabled monitors are started. Each monitor first creates DFM host groups for itself and both its sites using DFM command line operations. If the groups already exist, any existing members of the groups are first removed, and then the controllers defined at each site are added. This keeps the group contents current should the controllers assigned to each site change.

The status of the service can be checked and the service started and stopped using normal service monitoring commands (e.g. `/etc/init.d/NTAPmctb status` (Unix/Linux), `service mctb` (Windows)), or through the DFM plugin script. The DFM plugin script can also be used to list and view the available log files.

Script usage information can be found in Operations Manager by navigating to **Management->Scripts**, selecting the *MetroCluster TieBreaker* script name link to arrive at SCRIPT DETAILS screen, and selecting the question mark (?) next to the script title (see the *STEP 1 – Install the Operations Manager script plugin* above for screen examples).

7.1 EVENTS AND ALERTING

The MetroCluster TieBreaker issues DFM Events for various conditions it detects, or when it cannot perform its function properly (see the section on *Configuration-Failed* below). These events appear in Operations Manager and can be configured to generate Alarms. Alarms can be configured to send email or send SNMP Traps to other systems. For more information on configuring Alarms, please refer to [*Operations Manager Administration Guide For Use with DataFabric® Manager Server 4.0.*](#)

8 MONITORING PROCEDURE

Each monitor defined in the configuration file operates independently and monitors both controllers in the MetroCluster. The polling interval configured for the monitor determines how long the monitor will pause between each monitoring activity. The monitoring activity itself takes time, so the total time from the start of one iteration to the next is the polling interval plus the time it takes for the monitor to test the environment.

The following steps are conducted, in order, for each iteration of the monitor:

8.1 SITE REACHABILITY TESTING

If gateways are configured for a site, reachability is tested by attempting to ping each gateway. First site0 is tested, then site1.

If multiple gateways are defined for a site, the gateways are tested in parallel and the test completes when the first gateway responds, or all gateways have timed out after the configured number of retries (see **pingTimeout** and **pingRetries** above). If neither site can be reached, a DFM event is issued (see **Isolation-Detected**, below). Site isolation prevents any further testing and failover is disabled for that monitor until communications with a site is restored.

If an outstanding Isolation-Detected event exists for a monitor and, during a subsequent iteration, one or more gateways become reachable, an **Isolation-Resolved** event is issued, clearing the Isolation-Detected event.

8.2 FILER STATUS UPDATE

Next, cluster status information is gathered from the controller at each site. First the controller at site0 is checked, then the controller at site1. If cluster failover is not enabled, or either controller is in a takeover transitional state, no further testing is done and the test cycle completes.

During this operation the monitor communicates with each controller using ONTAP API. If communications fail, either an **Authentication-Failure** or an **Api-Error** event is issued. If one of these events is outstanding, and communications are restored in a subsequent iteration, a **Communication-Restored** event is issued, which clears the error events above (see event definitions below). Communication failures prevent any further testing and failover is disabled until communications are restored.

Next, cluster interconnect links are checked. If the link was previously up, and is listed as down by either controller, a **Link-Down** event is generated. If the interconnect link is up, and was previously down, a **Link-Up** event is generated, clearing any previous Link-Down event.

Finally, aggregate status is gathered. This information is used during CFOD checking and to start a mirror degradation timer.

8.3 CFOD CHECK

This check determines if a CFOD should be initiated. First the controller at site0 is checked, then the controller at site1. A CFOD is initiated under the following conditions:

- The controller is not reachable
- The controller's partner has a cf status of WAITING or ERROR
- The controller's partner shows that the interconnect link is down
- The aggregate containing the root volume is mirror degraded
- No un-ignored aggregate has exceeded the *cfodAbortTimeout* threshold.

If all other conditions exist, but an aggregate has exceeded the *cfodAbortTimeout*, a **CFOD-Aborted** event is thrown (see event definitions below).

8.4 MIRROR DEGRADATION CHECK

If CFOD conditions have not been met, un-ignored aggregates are checked for mirror degradations. If any mirrors are degraded, a timer is started and a **Mirror-Degradation** event is thrown (see event definitions below). If any mirrors already have active degradation timers, and a timer has exceeded the *cfodAbortTimeout*, a **Threshold-Exceeded** event is thrown. If all mirrors previously degraded are found to be in sync, a **Mirror-Normal** event is thrown, which clears any Mirror-Degradation or Threshold-Exceeded events.

8.5 SITE FAILOVER

If the CFOD Check described above detects a failover condition, a forced takeover is issued to the surviving controller (*cf forcetakeover -d*). The monitor then waits asynchronously for CFOD status, polling the surviving controller every 5 seconds, until either a success or failure status is received or *cfodSuccessWaitTimeout* has been exceeded. If a status of TAKEOVER is found, a **CFOD-Succeeded** event is thrown. If a status of TAKEOVER_FAILED is found, a **CFOD-Failed** event is thrown. Any other status will continue the polling. If the *cfodSuccessWaitTimeout* is exceeded, a **CFOD-Unknown** event is thrown and polling stops.

9 DFM EVENTS

The following events can be issued by MetroCluster TieBreaker.

9.1 AUTHENTICATION-FAILURE

Severity: Error

Class: MetroCluster-TieBreaker:Communications-Event

This event is thrown when MCTB fails to authenticate with a controller. This can happen when the controller's credentials have been changed but the MCTB configuration has not been updated. To correct the problem, update the credentials in the MCTB configuration file and restart MCTB.

9.2 API-ERROR

Severity: Error

Class: MetroCluster-TieBreaker:Communications-Event

This event is thrown when MCTB fails to invoke an ONTAP API call on a controller. This usually indicates that the controller is not reachable or that http admin access has been disabled (see **options http.admin.enable**).

9.3 COMMUNICATION-RESTORED

Severity: Normal

Class: MetroCluster-TieBreaker:Communications-Event

This event is thrown when a controller has either an outstanding Authentication-Failure or Api-Error event, and communications with the controller are restored. This event resets any outstanding Authentication-Failure or Api-Error event for the controller.

9.4 CONFIGURATION-FAILED

Severity: Critical

Class: MetroCluster-TieBreaker:Configuration-Error

This event is thrown on MCTB startup when the configuration file cannot be parsed, or when required values are not present or are not of the correct format. Check MCTB logs for failure details. To correct the problem, fix the configuration issues identified in the log file and restart MCTB.

This event must be acknowledged and/or deleted manually; no resetting event exists.

9.5 ISOLATION-DETECTED

Severity: Critical

Class: MetroCluster-TieBreaker:Isolation-Event

This event is thrown when none of the gateways identified at either site, for a given monitor, can be reached. This usually indicates that the MCTB hosting site has lost network or internet connectivity, and does not necessarily indicate a disaster at a monitored site. This condition prevents MCTB from further monitoring, so no failover can take place while this condition persists.

9.6 ISOLATION-RESOLVED

Severity: Normal

Class: MetroCluster-TieBreaker:Isolation-Event

This event is thrown when a monitor has an outstanding Isolation-Detected event, and one or more getaways at either site become reachable. This event resets an outstanding Isolation-Detected event for the monitor.

9.7 LINK-DOWN

Severity: Critical

Class: MetroCluster-TieBreaker:Interconnect-Event

This event is thrown when the cluster status returned by a controller indicates that the cluster interconnect is down.

9.8 LINK-UP

Severity: Normal

Class: MetroCluster-TieBreaker:Interconnect-Event

This event is thrown when there is an outstanding Link-Down event, and the status returned by the controller indicates the link is up. This event clears any existing Link-Down events.

9.9 MIRROR-DEGRADED

Severity: Error

Class: MetroCluster-TieBreaker:Degradation-Event

This event is thrown when an un-ignored aggregate is mirror degraded (and a CFOD condition does not also exist).

9.10 THRESHOLD-EXCEEDED

Severity: Critical

Class: MetroCluster-TieBreaker:Degradation-Event

This event is thrown when the mirror degraded timer has reached or exceeded the *cfodAbortTimeout*. Automatic CFOD is disabled as long as this condition persists.

9.11 MIRROR-NORMAL

Severity: Normal

Class: MetroCluster-TieBreaker:Degradation-Event

This event is thrown when there is an outstanding mirror degradation event and the mirrors are no longer degraded. This event resets any outstanding Mirror-Degraded or Threshold-Exceeded events.

9.12 CFOD-SUCCEEDED

Severity: Critical

Class: MetroCluster-TieBreaker:CFOD-Result

This event is thrown when CFOD has been initiated, and the result was successful (the surviving controller returned a status of TAKEOVER).

9.13 CFOD-FAILED

Severity: Critical

Class: MetroCluster-TieBreaker:CFOD-Result

This event is thrown when CFOD has been initiated, and the result was unsuccessful (the surviving controller returned a status of TAKEOVER_FAILED).

9.14 CFOD-UNKNOWN

Severity: Critical

Class: MetroCluster-TieBreaker:CFOD-Result

This event is thrown when CFOD has been initiated, and the no result was seen before *cfodSuccessWaitTimeout* expired (the surviving controller did not return a status of TAKEOVER or TAKEOVER_FAILED).

9.15 CFOD-ABORTED

Severity: Critical

Class: MetroCluster-TieBreaker:CFOD-Result

This event is thrown when the conditions called for a CFOD, but *cfodAbortTimeout* had been exceeded.

APPENDIX A – CONFIGURATION FILE EXAMPLE

```
<configuration version="1.0">
  <monitor name="MetroCluster MC1">
    <enabled>true</enabled>
    <pollingInterval>20</pollingInterval>
    <cfodAbortTimeout>90</cfodAbortTimeout>
    <cfodSuccessWaitTimeout>120</cfodSuccessWaitTimeout>
    <testMode>true</testMode>
    <site0 name="SiteA">
      <pingTimeout>5</pingTimeout>
      <pingRetries>3</pingRetries>
      <filer name="FAS3070-SITEA">
        <hostname>10.61.132.10</hostname>
        <username>root</username>
        <encryptedPassword>vi-Ld4yaFdyMf8SIgPQVFA</encryptedPassword>
        <ssl>false</ssl>
        <connectTimeout>10</connectTimeout>
        <connectRetries>1</connectRetries>
        <ignoredAggrs>
          <ignoredAggr>ignored_aggr</ignoredAggr>
        </ignoredAggrs>
      </filer>
      <gateways>
        <gateway>10.61.132.1</gateway>
      </gateways>
    </site0>
    <site1 name="SiteB">
      <pingTimeout>5</pingTimeout>
      <pingRetries>3</pingRetries>
      <filer name="FAS3070-SITEA">
        <hostname>10.61.132.11</hostname>
        <username>root</username>
        <encryptedPassword>vi-Ld4yaFdyMf8SIgPQVFA</encryptedPassword>
        <ssl>false</ssl>
        <connectTimeout>10</connectTimeout>
        <connectRetries>1</connectRetries>
      </filer>
      <gateways>
        <gateway>10.61.132.1</gateway>
      </gateways>
    </site1>
  </monitor>
</configuration>
```

APPENDIX B – ONTAP API (ONTAPI) CALLS USED

The following is a list of all the ONTAPI API calls used. The user configured for each Filer must be assigned to groups with roles that have the capabilities shown.

ONTAP API/CLI Command	Capability Required
ONTAPI Access	login-http-admin
aggr-list-info	api-aggr-list-info
cf-force-takeover	api-cf-force-takeover
cf-status	api-cf-status
system-get-info	api-system-get-info
volume-container	api-volume-container
volume-get-root-name	api-volume-get-root-name
volume-list-info	api-volume-list-info

As an example, the following controller CLI command creates a role containing all the necessary capabilities for this application:

```
useradmin role add MCTBRole -a login-http-admin,api-aggr-list-info,api-cf-force-takeover,api-cf-status,api-system-get-info,api-volume-container,api-volume-get-root-name,api-volume-list-info
```