

# Neuron Utilities

USER'S GUIDE

Product Version: 1.16.0

Document Revision: 1.0.0



## Copyright

Copyright © 1995-2011 Halcyon Monitoring Solutions, Inc.  
All rights reserved. This product and related documentation is protected by copyright and distributed under licenses restricting its use, copying, distribution and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Halcyon Monitoring Solutions, Inc. and its licensors.

Corporate Headquarters  
Halcyon Monitoring Solutions  
800 Bellevue Way NE,  
Suite 400  
Bellevue, WA 98004, USA

Tel: 416-932-4600  
Fax: 416-932-4711  
Email: [info@HalcyonInc.com](mailto:info@HalcyonInc.com)  
URL: [www.HalcyonInc.com](http://www.HalcyonInc.com)

## License Agreement

Downloading Halcyon software constitutes acceptance of the End User Binary Code License agreement, which can be found here: <http://www.halcyoninc.com/products/license.php>

Without purchasing a license, the Halcyon Software will only operate for a trial period of 14 days from installation. If you wish to purchase a license to use the Halcyon Software, please contact us at:

Web: <http://www.HalcyonInc.com>  
Email: [info@HalcyonInc.com](mailto:info@HalcyonInc.com)  
Tel: 416-932-4647  
Fax: 416-932-4711

## Technical Support

For assistance with any Halcyon products, contact Technical Support:

Tel: 1-877-932-4666 (Toll Free in North America)  
Tel: 1-416-932-4666 (International)  
Email: [support@HalcyonInc.com](mailto:support@HalcyonInc.com)

### Halcyon Forums:

Halcyon experts actively participate in the online Halcyon Forums. The experts are constantly monitoring the forums, answering questions and posting useful tips, tricks, and general knowledge base information. Whether you have a technical question or just wish to expand our knowledge base, this is the place for you. <http://forums.HalcyonInc.com>

## About Halcyon

Halcyon delivers Infrastructure Management solutions that provide operational visibility, availability, and reliability for business critical services and their underlying infrastructure. Since 1994, numerous Fortune 100 and SMEs, spanning every major geography and sector, have adopted Halcyon solutions.

At Halcyon, we believe the health of the IT infrastructure is integral to the success of a business. Our clients rely on us for complete end-to-end monitoring solutions that are straightforward, easy to deploy and use, and cost-effective, coupled with a history of client service excellence.

### **Your Infrastructure is Our Business**

# Contents

---

<b>1</b>	<b>PREFACE</b>	<b>6</b>
	1.1 PURPOSE OF THE DOCUMENT	6
	1.2 INTENDED AUDIENCE	6
	1.3 RELATED DOCUMENTS	6
<b>2</b>	<b>GENERAL OVERVIEW</b>	<b>7</b>
	2.1 UTILITIES OVERVIEW	7
	2.2 DEFINITIONS	7
	2.2.1 BASEDIR	7
	2.2.2 LOCALDIR	7
	2.2.3 USER	8
	2.2.4 PASSWORD	8
<b>3</b>	<b>CANCELLING PENDING ACTIONS</b>	<b>9</b>
<b>4</b>	<b>PROCESS RULES</b>	<b>10</b>
	4.1 EXAMPLES	11
	4.2 FILE COMMAND	12
<b>5</b>	<b>PROCESS ASSET</b>	<b>13</b>
	5.1 SUPPORTED COMMANDS	13
	5.2 EXAMPLES	14
<b>6</b>	<b>PROCESS ASSET CREDENTIALS</b>	<b>15</b>
	6.1 DETERMINE THE ASSET TO ACT ON	15
	6.2 DETERMINE THE CREDENTIALTYPE	15
	6.3 ADD, UPDATE, REMOVE OR COPY THE CREDENTIAL	16
	6.4 IMPORTANT NOTES	16
<b>7</b>	<b>PROCESS TAG INFO (EVENT TAGS)</b>	<b>17</b>
	7.1 LISTTAGS	17
	7.1.1 EXAMPLES	17
	7.2 GETTAGINFO	18
	7.2.1 EXAMPLES	18
	7.3 ADDTAG	18
	7.3.1 EXAMPLES	18
	7.4 REMOVETAG	18
	7.4.1 EXAMPLES	19
	7.5 IMPORTFROMFILE	19

7.5.1	NOTES	19
7.5.2	EXAMPLES	19
<b>7.6</b>	<b>EXPORTTOFILE</b>	<b>20</b>
7.6.1	EXAMPLES	20
<b>7.7</b>	<b>BULKFILEREMOVE</b>	<b>20</b>
7.7.1	EXAMPLES	21
<b>8</b>	<b>PROCESS SNAPSHOT</b>	<b>22</b>
<b>8.1</b>	<b>SNAPSHOT CONFIGURATION FILE</b>	<b>22</b>
8.1.1	SNAPSHOT CONFIGURATION	22
8.1.2	SCHEDULE CONFIGURATION	23
8.1.3	REPORT CONFIGURATION	24
<b>8.2</b>	<b>CREATING AND LISTING SNAPSHOTS</b>	<b>25</b>
<b>8.3</b>	<b>COMPARING SNAPSHOTS</b>	<b>25</b>
<b>8.4</b>	<b>SCHEDULING SNAPSHOTS</b>	<b>26</b>
<b>8.5</b>	<b>LISTING AND DELETING SCHEDULES</b>	<b>26</b>
<b>8.6</b>	<b>EXAMPLES</b>	<b>27</b>
<b>9</b>	<b>PROCESS REVISION HISTORY EXPORT</b>	<b>28</b>
<b>9.1</b>	<b>REQUIRED PARAMETERS</b>	<b>28</b>
<b>9.2</b>	<b>OPTIONAL PARAMETERS</b>	<b>28</b>
<b>9.3</b>	<b>BASIC EXAMPLE</b>	<b>28</b>
<b>9.4</b>	<b>TARGETED EXAMPLES</b>	<b>29</b>
<b>9.5</b>	<b>NOTES</b>	<b>29</b>
<b>10</b>	<b>EXECUTE OPERATION</b>	<b>30</b>
<b>10.1</b>	<b>SUPPORTED PARAMETERS</b>	<b>30</b>
<b>10.2</b>	<b>EXAMPLES</b>	<b>30</b>
<b>10.3</b>	<b>COMMAND SERVICES</b>	<b>31</b>
<b>10.4</b>	<b>HOW IT WORKS</b>	<b>31</b>
<b>11</b>	<b>MESSAGE GATEWAY</b>	<b>32</b>
<b>11.1</b>	<b>SUPPORTED PARAMETERS</b>	<b>32</b>
<b>11.2</b>	<b>INJECTING MULTIPLE MESSAGES</b>	<b>32</b>
<b>11.3</b>	<b>EXAMPLES</b>	<b>33</b>

## Tables

---

TABLE 1.2-1: INTENDED AUDIENCE	6
TABLE 1.3-1: RELATED DOCUMENTS	6

# 1 Preface

## 1.1 Purpose of the Document

The purpose of this document is to describe the various command line utilities available for the Halcyon Neuron Management Suite and how to use them.

## 1.2 Intended Audience

This guide is written for the following type of audience:

**Table 1.2-1: Intended Audience**

Role	Usage
End User	The User's Guide is intended for end users who use the product on a daily basis. This guide provides information on how to use the product for tasks such as generating and viewing reports.
Manager	The User's Guide provides information for managers who are responsible for preparing the product for use by end users. Their tasks include the setup of graph templates, report templates, and report schedules.
Administrator	The User's Guide contains details information for administrators in order to configure the product.

## 1.3 Related Documents

The Halcyon Neuron Management Suite is composed of a series of underlying products. For further information regarding the configuration, usage and administration of these products, please refer to the following documents.

These documents may be located in the doc folder of the solution distribution or on the website ([www.halcyoninc.com/docs](http://www.halcyoninc.com/docs)).

**Table 1.3-1: Related Documents**

Component Name	Related Documents
Neuron Management Suite	<ul style="list-style-type: none"> <li>▪ Neuron Management Suite Installation Guide</li> </ul>
Neuron Management Server	<ul style="list-style-type: none"> <li>▪ Neuron Management Server User's Guide</li> <li>▪ Neuron Management Server Release Notes</li> </ul>
Neuron Performance Manager	<ul style="list-style-type: none"> <li>▪ Neuron Performance Manager Release Notes</li> </ul>

## 2 General Overview

---

### 2.1 Utilities Overview

This guide describes the following utilities that are part of the Halcyon Neuron Management Suite distribution:

- Cancel Pending Actions
- Process Rules
- Process Asset
- Process Asset Credentials
- Process Tag Info (Event Types)
- Process Revision History Export
- Process Snapshot
- Execute Operation
- Message Gateway

### 2.2 Definitions

#### 2.2.1 BASEDIR

The base directory where the Halcyon Neuron Management Suite was installed. The default value is `/opt/HMF`. To determine the value of `BASEDIR`, run one of the following commands.

On a Solaris host (for Solaris 10+ zoned environment, this must be a non-sparse root zone):

```
# pkgparam HALhmfcom BASEDIR
```

On a Linux host:

```
# rpm -q --queryformat '%{INSTALLPREFIX}\n' HALhmfcom
```

#### 2.2.2 LOCALDIR

The configuration and logfile directory of the Halcyon Neuron Management Suite. By default this is `/var/opt/HMF`. To determine `LOCALDIR`, run one of the following commands:

On a Solaris host (for Solaris 10+ zoned environment, this must be a non-sparse root zone):

```
# pkgparam HALhmfcom LOCALDIR
```

On a Linux host:

```
# cat [BASEDIR]/globalInstallVariables | grep LOCALDIR
```

### 2.2.3 USER

The name of the user running the utility. It must be defined in [LOCALDIR]/conf/users.xml.

### 2.2.4 PASSWORD

The password of the user running the utility. If the utility requires a password the user is requested to enter it on the command-line.

## 3 Cancelling Pending Actions

---

If any of the Neuron Rules specify an action that is executed repeatedly then Neuron Management Suite will execute the action at the specified interval until a close or acknowledge event for the open event occurs. In order to cancel these pending actions execute the following commands:

```
% su -  
# [BASEDIR]/bin/CancelPendingActions -user USER
```

This will cancel all pending actions that are currently queued in Neuron Management Suite.

## 4 Process Rules

---

The Neuron ProcessRule utility can be used to manipulate rules stored in database directly:

- to list information of rules in a rule base,
- to enable a particular rule or all rules in a rule base,
- to disable a particular rule or all rules in a rule base,
- to add a rule,
- to add/update a filter to a rule,
- to remove a filter from a rule,
- to remove a rule, which actually marks the rule as deleted,
- to undelete rules that have been marked as deleted,
- to purge the rules marked as deleted from database.

These can be done individually via the command line, or in bulk operation via a source file.

Please note that it is not possible to add, update, or remove rule schedules and policies. Filters apply to the EventManager rule base only.

The commands supported by this utility are:

-help	This will display the usage of this utility, including a list of all possible arguments and options, and examples.
-listAll	This will list attributes of a specified rule or attributes of all rules in a specified rule base, including rules that have been marked as deleted. If rule base is not specified, the default rule base is EventManager.
-listDeleted	This will list attributes of all rules that have been marked as deleted in a specified rule base.
-enable	This will enable a particular rule or all rules in a specified rule base.
-disable	This will disable a particular rule or all rules in a specified rule base.
-add	This will add a new rule with/without a filter to the database, or add a filter to an existing rule.
-update	This will update a filter of an existing rule. The filter attribute name need to be specified.
-remove	This will either mark a rule as deleted if only rule name is specified, or delete a filter from a rule if a filter attribute name is also specified.
-undelete	This will mark rule(s) that have been marked as deleted back to the state as not being deleted.
-purge	This will physically remove rule(s) that have been marked as deleted from database.
-file	This will process multiple commands defined in a source file.

Upon successful completion of the "listAll" or "listDeleted" command, the requested rules' information is displayed in list form. Use this information as options to the other commands.

Upon completion of the other commands a status message is displayed indicating whether the operation was successful or not.

To determine the filter attribute name, run this utility with the '-listAll' command to list all attributes of a rule:

```
[BASEDIR]/bin/ProcessRule -user USER -listAll
                        -ruleBase EventManager -rule "Rule 102"
```

The result will be a table like the following:

Rule Base	Rule	Attribute	Value
EventManager	Rule 102	name	Rule 102
EventManager	Rule 102	lastModified	2009-10-26 09:22:39.480000 GMT-0400
EventManager	Rule 102	enabled	true
EventManager	Rule 102	deleted	false
EventManager	Rule 102	lastRun	2009-10-26 11:14:15.684000 GMT-0400
EventManager	Rule 102	lastModifiedBy	system
EventManager	Rule 102	schedule.0	Hour < 23
EventManager	Rule 102	filter.0	where host contains "hst1" or "hst2"
EventManager	Rule 102	filter.1	where message contains "m1" or " m2"
EventManager	Rule 102	filter.2	where proxy type is "prxyT" or "pT2"
EventManager	Rule 102	filter.3	where message not contains "msg x"
EventManager	Rule 102	filter.4	where severity is CRITICAL
EventManager	Rule 102	schedule.0	Hour < 22
EventManager	Rule 102	schedule.1	Hour > 6
EventManager	Rule 102	action.0.description	Send email if Rule 102 is matched
EventManager	Rule 102	action.0.name	Rule 102 Action 0
EventManager	Rule 102	action.0.action	Send event
EventManager	Rule 102	action.0.parameter	to email adapter
EventManager	Rule 102	action.0.parameter	from "Halcyon Email Adapter"
EventManager	Rule 102	action.0.parameter	for recipients myName@myCompany.com
EventManager	Rule 102	action.0.parameter	with subject "Event on %asset - ....
EventManager	Rule 102	action.0.parameter	in html format

Find the filter attribute name that you want to update or remove (e.g., "filter.3"), and pass it to the -update or -remove command.

## 4.1 Examples

For a full list of examples, run this utility with -help command.

```
ProcessRule -user admin -listAll
ProcessRule -user admin -listAll -rule "Rule YY"
ProcessRule -user admin -listDeleted
ProcessRule -user admin -listDeleted -ruleBase "EventManager"
ProcessRule -user admin -enable
ProcessRule -user admin -enable -ruleBase "EventManager" -rule "Rule A"
ProcessRule -user admin -disable
```

```
ProcessRule -user admin -disable
            -ruleBase "EventManager" -rule "Rule B"

ProcessRule -user admin -add
            -ruleBase "EventManager" -rule "Rule A"
            -description "Description for Rule A" -category "Category A"

ProcessRule -user admin -add
            -rule "Rule A" -filter sourceHost contains "hostM" "hostN"
            -comment "Filter on source host name contains hostM or hostN"

ProcessRule -user admin -update -rule ruleC
            -description "A rule for neuron group" -category "Neuron group"
            -filterAttribute filter.2
            -filter sourceHost contains "hostXX" "hostYY"
            -comment "Filter on source host name contains hostXX or hostYY"

ProcessRule -user admin -remove
            -ruleBase "EventManager" -rule "rule A"

ProcessRule -user admin -remove
            -rule "rule D" -filterAttribute filter.0

ProcessRule -user admin -undelete
            -ruleBase "EventManager" -rule "rule A" "rule B" "rule C"

ProcessRule -user admin -undelete

ProcessRule -user admin -purge
            -ruleBase "EventManager" -rule "rule X" "rule Y" "rule Z"

ProcessRule -user admin -purge

ProcessRule -user admin -file addRulesRequests.txt
```

## 4.2 File Command

For -file command, the source file must have the following format:

Each line starts with one of the following commands:

-listAll, -listDeleted, -enable, -disable, -add, -update, -remove, -undelete, -purge  
Followed by its required optional arguments.

Run the utility with -help command for a full list of required/optional arguments.

## 5 Process Asset

The Neuron Process Asset utility can be used to list objects that are stored in Neuron Management Suite's database. It also supports adding and removing of certain objects as well as enabling/disabling data collection for Neuron agents.

In order to list and add/remove database objects or to enable/disable data collection for Neuron agents execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessAsset -user USER COMMAND OPTIONS
```

### 5.1 Supported Commands

This utility supports the following commands:

-list	This will list objects in the database. In order to select what type of objects are returned the command supports several options. Execute the command with the -help option to display a list of all possible options.
-add	This will add an object to the database.  Host and port are only required when an asset of type "agent.neuron", or "agent.sunmc" is added.  Port is also required when an asset of type "bmc" (Baseboard Management Controller) is added.
-remove	This will remove an object from the database.
-enable	This will enable data collection for an asset in the database.  NOTE: Currently only supports types "agent.neuron" and "agent.sunmc".
-disable	This will disable data collection for an asset in the database.  NOTE: Currently only supports types "agent.neuron" and "agent.sunmc".
-export	This will export asset information from the database to an XML file.
-exportComplianceInventory	This will export an XML snapshot of all system assets from the cmdb along with relevant information known about the asset (ie. os version). This command is run with the -file parameter specifying the XML file to export the inventory to.
-convert	This will convert asset information contain in an XML file to CSV (option -xmlIn). The resulting CSV file can also be converted into XML (option -csvIn).

-changeHostname	<p>This will change the hostname of an asset along with its Neuron and/or Sun MC agent if one exists. This command requires the -assetMonikerToUpdate and -newHostname parameters that specify an asset and a hostname respectively. If there is an agent on the system, you will also need to provide the agent's port. -newNeuronPort is required for updating systems with a Neuron Agent and -newSunMCPort is required for systems with a Sun MC Agent.</p> <p>NOTE: The asset given as the assetMonikerToUpdate must be of assetType "system".</p> <p>NOTE: The Neuron Port needs to be the Neuron Agent's http port.</p>
-----------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Upon successful completion of the list command the requested objects are displayed in list form. Use this information as options to the other commands.

Upon successful completion of the other commands a status message is displayed indicating whether the operation was successful or not.

## 5.2 Examples

```
ProcessAsset -user admin -list -assetType agent.neuron hmf
```

```
ProcessAsset -user admin -list -namespace hmf
```

```
ProcessAsset -user admin -add -asset myAsset myType other
```

```
ProcessAsset -user admin -enable
             -asset myHost.6680 agent.neuron hmf
```

```
ProcessAsset -user admin -export -r -assetType system hmf
             -file /var/opt/HMF/assetExport.xml
```

```
ProcessAsset -user admin -exportComplianceInventory
             -file /var/opt/HMF/inventoryReport.xml
```

```
ProcessAsset -user admin -convert
             -xmlIn /var/opt/HMF/assetExport.xml -out /var/opt/HMF/assets.csv
```

```
ProcessAsset -user admin -convert
             -csvIn /var/opt/HMF/assets.csv -out /var/opt/HMF/assets.xml
```

## 6 Process Asset Credentials

In order to add a new credential (username and password) to an asset, update a credential of an asset, remove a credential from an asset, or copy a credential from one asset to another, you will first need to collect some information.

### 6.1 Determine the Asset to Act on

To do any credential operation on an asset, you must first determine the asset (host, agent, etc) to do that operation on. The asset is defined by a name, a type and a namespace, all of which must be provided as parameters to the utility. To determine these values, run this utility to list out the available host assets:

```
[BASEDIR]/bin/ProcessAssetCredentials -user USER -listHosts
```

The result will be a table like the following:

Host Name	Port	Asset	Type	Namespace
toronto	6680	toronto.6680	agent.neuron	hmf
atlantis	6680	atlantis.8800	agent.neuron	hmf
discovery	6680	discovery.6680	agent.neuron	hmf

Find the Host Name you want to do the credential operation on, then take note of its Asset, Type and Namespace.

If you are going to copy a credential from one asset to another, also take note of the Asset, Type and Namespace of the asset you are copying the credential from (sourceAsset, sourceType, etc).

### 6.2 Determine the CredentialType

If you are adding or updating a credential you should know what the credential type is (SSH, WMI or AGENT\_NEURON) based on what you are adding or updating the credential for (see associated documentation). Use this credential type as the credentialType parameter for the utility.

If you are going to remove or copy a credential from an asset, you can determine the type of the credential you are going to act on by listing out the users for that asset. Run the utility using the list users command and the asset (for removal) or sourceAsset (for copy) details noted in 6.1 above:

```
[BASEDIR]/bin/ProcessAssetCredentials -user USER -listUsers
-asset toronto.6680 agent.neuron hmf
```

The result will be a listing like the following:

Host Name	Port	Asset	Type	Nsp	Users
toronto	6680	toronto.6680	agent.neuron	hmf	admin (AGENT_NEURON) admin (SSH)

Take note of the user you wish to act on and its type.

## 6.3 Add, Update, Remove or Copy the Credential

Having gathered all the details in steps 6.1 and 6.2 above, you can then run the utility with the `-add`, `-update`, `-remove` or `-copy` command specifying the data from above as parameters.

For instance, if you wanted to add user `robert` with password `halcyon` to asset `atlantis.8800` so it could access a neuron agent, you would run the utility as follows:

```
[BASEDIR]/bin/ProcessAssetCredentials -user admin -add
  -asset atlantis.8800 agent.neuron hmf
  -credential AGENT_NEURON robert halcyon
```

The result will be something like:

```
Credential 'robert' was added successfully.
```

As another example, if you wanted to add user `robert` with password `halcyon` to asset `robertHome` so that it can access the asset using WMI, you would run the utility as follows:

```
[BASEDIR]/bin/ProcessAssetCredentials -user admin -add
  -asset robertHome system hmf -credential WMI robert halcyon localhost
```

Where `'localhost'` is the domain of user `robert` on the WMI com server.

Alternatively, you could copy the `ssh admin` user from `toronto.6680` to `discover.6680` by running the utility as follows:

```
[BASEDIR]/bin/ProcessAssetCredentials -user admin -copy
  -asset discover.6680 agent.neuron hmf
  -sourceAsset toronto.6680 agent.neuron hmf -credential SSH admin
```

## 6.4 Important Notes

The credential password is not required for copy or remove commands.

If a user already exists as a credential for an asset, running the add command will not update its password.

If a user does not exist as a credential for an asset, running the update command will add its password.

The credential domain is required only for add and update commands for WMI credentials. If provided for other credential types, it will be ignored.

---

## 7 Process Tag Info (Event Tags)

---

Event Tags are tags that have been added to the database with a specific pattern. The tag is associated with an Event as an Event Tag if the Event matches that pattern.

The ProcessTagInfo utility allows you to get information about tags in general and Event Tags specifically. It also allows you to add and delete tags, either individually via the command line, or in bulk via a source file.

The commands available via the ProcessTagInfo utility are as follows:

- listTags
- getTagInfo
- addTag
- removeTag
- importFromFile
- bulkFileRemove
- exportToFile

The ProcessTagInfo utility is run as follows:

```
[BASEDIR]/bin/ProcessTagInfo -user USER COMMAND <PARAMS>
```

To get help in general:

```
[BASEDIR]/bin/ProcessTagInfo -help
```

To get help with a specific command (ie. the -addTag command):

```
[BASEDIR]/bin/ProcessTagInfo -help -addTag
```

### 7.1 listTags

The “-listTags” command is used to list out all tags associated with a specific tag type. No parameters need to be provided for this command. By default tags being retrieved are those associated with the 'event' tag type. If you wish to list out tags associated with another type, provide the -tagTypeName parameter.

#### 7.1.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -listTags
```

```
[BASEDIR]/bin/ProcessTagInfo -user admin -listTags -tagTypeName miscType
```

## 7.2 getTagInfo

The “-getTagInfo” command is used to get all the details about a single tag (as specified on the command line) from the database. The -tagName parameter is the minimum that is required. By default tags being retrieved are those whose type is 'event'. If you wish to get information about a tag with a different type, provide the -tagTypeName parameter.

### 7.2.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -getTagInfo  
    -tagName sampleTag
```

```
[BASEDIR]/bin/ProcessTagInfo -user admin -getTagInfo  
    -tagName sampleTag -tagTypeName miscType
```

## 7.3 addTag

The “-addTag” command is used to add a single tag (as specified on the command line) into the database or update a tag that already exists. The -tagName parameter is the minimum that is required.

-tagDescription, -tagPattern, -tagNotes can be provided, and should be in this is a new tag being added.

For updating an existing tag, the fields to update (-tagDescription, -tagPattern, -tagNotes) should be provided. To change the name of a tag, add a new tag.

By default tags are added/updated as tags of type 'event'. If you wish to add a tag with a different type, provide the -tagTypeName parameter.

### 7.3.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -addTag  
    -tagName sampleTag -tagDescription "New tag added"  
    -tagPattern .*abc.*
```

## 7.4 removeTag

The “-removeTag” command is used to remove a single tag (as specified on the command line) from the database.

The -tagName parameter is the minimum that is required.

By default tags being removed are those whose type is 'event'. If you wish to remove a tag with a different type, provide the `-tagTypeName` parameter.

### 7.4.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -removeTag -tagName sampleTag
```

## 7.5 importFromFile

The “importFromFile” command is used to add/update multiple tags (as defined in a source file) from the database at once.

The `-sourceFile` parameter (path to the source file) is the minimum that is required.

By default tags being added/updated are those whose type is 'event'. If you wish to add/update tags with a different type, provide the `-tagTypeName` parameter.

The sourceFile file must have the following format:

Tags in the file must be separated by a line with a single dash '-'. The tag itself is composed of several lines, where 4 are required. Each of the name, description, pattern and notes begin on a line that starts the '>' as follows:

```
>TagName
>TagDescription
TagDescription continued
>TagPattern
>TagNotes
-
>....
```

### 7.5.1 Notes

Tag notes must not start with the '-' or '>' characters.

If there are problems with the file (such as have incorrect number of lines starting with '>' for a single tag) then no tags will be added. Import of tags will only be attempted if the file is properly formatted.

A sample import file can be found here:

```
[LOCALDIR]/conf/SampleTagImportFile
```

### 7.5.2 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -importFromFile
-sourceFile /var/opt/HMF/tagImport
```

```
[BASEDIR]/bin/ProcessTagInfo -user admin -importFromFile  
-sourceFile /var/opt/HMF/tagImport -tagTypeName miscType
```

## 7.6 exportToFile

The “-exportToFile” command is used to write out the details of all tags associated with a specific tag type to a designated file.

The only parameter that is required is the -destinationFile which must be a path to a file where the tag details will be written.

If the destinationFile does not exist, an attempt will be made to create it.  
If the destinationFile does exist, its content will be overwritten.

By default tags being retrieved are those associated with the 'event' tag type. If you wish to get tag information for tags associated with another type, provide the -tagTypeName parameter.

### 7.6.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -exportToFile  
-destinationFile /var/opt/HMF/tagOutput
```

```
[BASEDIR]/bin/ProcessTagInfo -user admin -exportToFile  
-destinationFile /var/opt/HMF/tagOutput -tagTypeName miscType
```

## 7.7 bulkFileRemove

The “-bulkFileRemove” command is used to remove multiple tags (as defined in a source file) from the database at once.

The -sourceFile parameter (path to the source file) is the minimum that is required.

By default tags being removed are those whose type is 'event'. If you wish to remove a tag with a different type, provide the -tagTypeName parameter.

The sourceFile file must be in a specific format as follows:

Tags in the file must be separated by a line with a single dash '-'.  
-

The tag itself is composed of several lines, where 4 are required. Each of the name, description, pattern and notes begin on a line that starts the '>' as follows:

```
>TagName  
>TagDescription  
TagDescription continued  
>TagPattern  
>TagNotes  
-  
>....
```

NOTE: Tag notes must not start with the '-' or '>' characters.

If there are problems with the file (such as have incorrect number of lines starting with '>' for a single tag) then no tags will be removed. Removal will only be attempted if the file is properly formatted.

### 7.7.1 Examples

```
[BASEDIR]/bin/ProcessTagInfo -user admin -bulkFileRemove  
-sourceFile /var/opt/HMF/tagDelete
```

## 8 Process Snapshot

Neuron Management Suite supports the creation of snapshots. These are collections of assets and their attributes and metrics taken from the database. Each snapshot is given a timestamp indicating the time the snapshot was taken. It is then stored in the directory `[LOCALDIR]/snapshots`. The following types of snapshots are supported:

- Configuration snapshots: The snapshot contains assets and their configuration metrics.
- Inventory snapshots: The snapshot contains assets only.

The Process Snapshot utility is used to perform the following functions:

- Create configuration and inventory snapshots
- Compare configuration and inventory snapshots
- Set up schedules for creating and comparing snapshots
- List and delete those schedules

### 8.1 Snapshot Configuration File

Each snapshot consists of one or more assets. In order to simplify the specification of these assets a separate configuration file is used. An example can be found in `[LOCALDIR]/conf/snapshots-example.xml`. Use this file as a starting point for specifying your own snapshot configurations. This file is also used to define schedules for creating or comparing snapshots as well as reports for comparing snapshots.

#### 8.1.1 Snapshot Configuration

The following fragment taken from the `snapshots-example.xml` file defines a snapshot.

```
<snapshot id="Systems" type="Inventory" name="Inventory Snapshot">
  <assetType moniker="system"/>
</snapshot>
```

The `snapshot` element supports the following attributes and elements:

id	An ID for the snapshot. This must be unique within the configuration file.
type	The type of snapshot. This must be either Configuration or Inventory.
name	The name for the snapshot.
asset	<p>The asset for which to create the snapshot. The following example specifies that the system <code>coolthreads</code> should be included in the snapshot:</p> <pre>&lt;asset moniker="coolthreads" assetType="system"/&gt;</pre> <p>Use one asset element for each asset that should be included in the snapshot.</p>

assetType	<p>The <code>asset</code> type for which to create the snapshot. The following example specifies that all system assets should be included in the snapshot:</p> <pre>&lt;assetType moniker="system" /&gt;</pre> <p>Use one <code>assetType</code> element for each asset type that should be included in the snapshot.</p>
-----------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Please note that elements `asset` and `assetType` may be specified together in the same snapshot configuration.

## 8.1.2 Schedule Configuration

The following fragment taken from the `snapshots-example.xml` file defines a snapshot.

```
<schedule name="Create Inventory Snapshot">
  <startTime>1970-01-01 00:00:00</startTime>
  <perform>every day</perform>
</schedule>
```

The `schedule` element supports the following attributes and elements:

name	A unique name in the configuration file.																
startTime	<p>The time when the first snapshot should be created. The following time format is supported:</p> <pre>&lt;YYYY-MM-DD HH:MM:SS&gt;</pre> <p>Where:</p> <table style="width: 100%; border: none;"> <tr> <td>YYYY – the year</td> <td>HH – the hour (00 – 23)</td> </tr> <tr> <td>MM – the month (01 – 12)</td> <td>MM – the minutes (00 – 59)</td> </tr> <tr> <td>DD – the day (01 – 31)</td> <td>SS – the seconds (00 – 59)</td> </tr> </table>	YYYY – the year	HH – the hour (00 – 23)	MM – the month (01 – 12)	MM – the minutes (00 – 59)	DD – the day (01 – 31)	SS – the seconds (00 – 59)										
YYYY – the year	HH – the hour (00 – 23)																
MM – the month (01 – 12)	MM – the minutes (00 – 59)																
DD – the day (01 – 31)	SS – the seconds (00 – 59)																
perform	<p>The interval at which the schedule is executed. It supports the following expressions:</p> <table style="width: 100%; border: none;"> <tr> <td>once</td> <td>every day</td> </tr> <tr> <td>every x seconds</td> <td>weekdays</td> </tr> <tr> <td>every minute</td> <td>every x days</td> </tr> <tr> <td>every x minutes</td> <td>every week</td> </tr> <tr> <td>every hour</td> <td>every x weeks</td> </tr> <tr> <td>every x hours</td> <td>day x</td> </tr> <tr> <td></td> <td>the (first   second   third   fourth   last)</td> </tr> <tr> <td></td> <td>(Monday   Tuesday   Wednesday   Thursday   Friday   Saturday   Sunday)</td> </tr> </table>	once	every day	every x seconds	weekdays	every minute	every x days	every x minutes	every week	every hour	every x weeks	every x hours	day x		the (first   second   third   fourth   last)		(Monday   Tuesday   Wednesday   Thursday   Friday   Saturday   Sunday)
once	every day																
every x seconds	weekdays																
every minute	every x days																
every x minutes	every week																
every hour	every x weeks																
every x hours	day x																
	the (first   second   third   fourth   last)																
	(Monday   Tuesday   Wednesday   Thursday   Friday   Saturday   Sunday)																

### 8.1.3 Report Configuration

The following fragment taken from the snapshots-example.xml file defines a report.

```
<report id="Summary" type="Configuration"
  stylesheet="configuration-summary-report.xsl">
  <title>Configuration Compliance: Summary Report</title>
  <name>Solaris Zone Compliance Report</name>
  <description>This is a quarterly report to verify the
    compliance level of Solaris Zones</description>
  <category>Configuration</category>
  <recipient>manager@localhost</recipient>
  <subject>%name - %timestamp</subject>
  <sender>neuron@localhost</sender>
  <format>html</format>
</report>
```

The following attributes and elements are used for defining a compliance report (see 8.3 Comparing Snapshots) that may be sent to one or more email recipients. To send a report to multiple recipients use one `recipient` element for each person that should receive a report. If no recipient is specified the report is not sent via email.

id	The ID of the report. This must be unique within the configuration file.
type	The type of report. This must be either "Configuration" or "Inventory" for summary report. Only "Configuration" type is supported for asset detail report and metric detail report.
stylesheet	The style sheet that is used to process the report. The following default style sheets for various reports are located at [LOCALDIR]/conf: <ul style="list-style-type: none"> <li>- configuration-asset-detail-report.xsl</li> <li>- configuration-metric-detail-report.xsl</li> <li>- configuration-summary-report.xsl</li> <li>- inventory-report.xsl</li> </ul> <p>You can provide your own style sheets. Note that the name of the style sheet for configuration asset detail report must contain "asset-detail-report"; and the name of the style sheet for configuration metric detail report must contain "metric-detail-report".</p>
title	The title of the report.
name	The name of the report.
description	The description of the report.
category	The category of the report.
recipient	The email address of each recipient of the report. The following formats are supported: <pre>user@domain full name &lt;user@domain&gt;</pre>
subject	The subject for the email. The parameters <code>%name</code> and <code>%timestamp</code> are placeholders for the name of the report and its creation time, respectively. If not specified the default is: <pre>%name - %timestamp</pre>

sender	The sender for the email. If not specified the default is: Neuron Compliance Manager <neuron@localhost>
format	The format for the email. This must be either <code>html</code> or <code>text</code> . If not specified, the default is: <code>html</code>

## 8.2 Creating and Listing Snapshots

In order to create a new snapshot, execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -create -snapshot <id> [-baseline]
```

The `file` specifies the complete path to the snapshot configuration file (see 8.1 Snapshot Configuration File).

The `id` is the snapshot ID defined in the snapshot configuration file. If `-baseline` is specified then the snapshot is marked as a baseline. This means it can be used to compare regular snapshots with this baseline snapshot (see 8.3 Comparing Snapshots).

For listing existing snapshots that have been created for a specific snapshot ID, execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -list -snapshot <id> [-baseline]
```

If `-baseline` is specified then only baseline snapshots will be listed. Without this argument all snapshots will be listed.

## 8.3 Comparing Snapshots

In order to compare two snapshots, execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -compare -snapshot <snapshotId> <snapshot> [<snapshot>]
  [-baseline [<baseline>]] -report <reportId>
```

The `file` specifies the complete path to the snapshot configuration file (see 8.1 Snapshot Configuration File).

The `snapshotId` is the snapshot ID defined in the snapshot configuration file. The snapshots that should be compared are specified with one or two `snapshot` arguments. Take these values from the output of listing snapshots (see 8.2 Creating and Listing Snapshots). If only one snapshot is specified then the parameter `-baseline` is implied and the snapshot is compared with the latest baseline.

Use `last` as value for `snapshot` in order to compare the latest snapshot with either another snapshot or a baseline.

Use `current` as value for `snapshot` in order to compare the current values with either another snapshot or a baseline.

If `-baseline` is specified then only one `snapshot` argument is required. The provided snapshot is then compared with the latest baseline.

If a baseline is specified it must be in the form `<asset moniker> <asset type> <namespace>`.

The `reportId` is the report ID defined in the snapshot configuration file.

When the command completes it prints the name of the generated report. If the report configuration includes recipients then the report is also sent to the provided email addresses.

## 8.4 Scheduling Snapshots

In order to schedule the creation or comparison of snapshots, execute the following commands:

For creating snapshots based on a schedule:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -create -snapshot <id> [-baseline]
  -schedule <name>
```

For comparing snapshots based on a schedule:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -compare -snapshot <snapshotId> <snapshot> [<snapshot>]
    [-baseline [<baseline>]] -report <reportId>
  -schedule <name>
```

The only difference here is the additional `-schedule` parameter. The argument `name` specifies the name of the schedule that can be found in the configuration file.

## 8.5 Listing and Deleting Schedules

In order to list existing snapshot schedules, execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
  -user USER
  -file <file>
  -list -schedule
```

The `file` specifies the complete path to the snapshot configuration file (see 8.1 Snapshot Configuration File).

This will print a list of all currently scheduled snapshots.

In order to delete an existing snapshot schedule, execute the following commands:

```
% su -
# [BASEDIR]/bin/ProcessSnapshot
   -user USER
   -file <file>
   -delete -snapshot <id> -schedule <name>
```

The `file` specifies the complete path to the snapshot configuration file (see 8.1 Snapshot Configuration File).

The `id` is the snapshot ID defined in the snapshot configuration file.

The `name` is the schedule name defined in the snapshot configuration file.

This will delete the schedule.

## 8.6 Examples

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -create -snapshot Systems
```

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -create -snapshot Systems -baseline
```

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -create -snapshot Systems
   -schedule "Create Inventory Schedule"
```

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -compare -snapshot Systems Inventory.1307969926595
   -baseline -report Summary
```

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -compare -snapshot Systems current
   -baseline host system hmf -report Summary
```

```
[BASEDIR]/bin/ProcessSnapshot -user admin
   -file <file> -compare -snapshot Hosts last
   -baseline -report Summary
   -schedule "Compare Configuration Schedule"
```

## 9 Process Revision History Export

Certain changes (mainly those that can be affected via the Neuron Management Portal (UI) or the aforementioned utilities such as ProcessAsset) that occur within the Neuron Management Suite database are tracked using revision history tables. By default these changes are logged to a revisionHistory.log file with all of the other Neuron Management Suite log files as they occur. However, it may be necessary to go back further in time to see changes made over a particular period of time; that is where this utility can be used.

The ProcessRevisionHistoryExport utility is run as follows:

```
[BASEDIR]/bin/ProcessRevisionHistoryExport -user USER COMMAND <PARAMS>
```

where currently the only available command is -export.

### 9.1 Required Parameters

- |            |                                                                                                  |
|------------|--------------------------------------------------------------------------------------------------|
| -logFile   | This is the full path to the file that the revision history should be exported to.               |
| -startTime | The date and time at which the revision data should start from (of the form YYYY/MM/DD:hh:mm:ss) |

### 9.2 Optional Parameters

- |           |                                                                                                                                                                                       |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -database | The database to log revision history for; valid values are CMDB and MIBDB (CMDB is default if none is provided).                                                                      |
| -endTime  | The date and time at which the revision data should end at (of the form YYYY/MM/DD:hh:mm:ss). If no endTime is provided, then the time the utility started execution at will be used. |

### 9.3 Basic Example

```
[BASEDIR]/bin/ProcessRevisionHistoryExport -user admin  
-export -logFile /var/opt/HMF/logs/myRevisions.log  
-startTime 2009/07/08:09:10:11 -endTime 2009/07/10:15:16:17
```

It is also possible to restrict the revisions logged to specific assets or types (such as show only changes to neuron agents or rules or when filters are changed). This may be useful if you would like to track all changes for something specific over a period of time.

To target revision logging, one or more of the following must be provided as parameters (can specify multiple times):

```
[-asset <asset> <type> <namespace>]
[-assetType <type> <namespace>]
[-attribute <attribute>]
[-tag <tagName> <tagTypeName>]
```

The following are the parameters that may be needed for targeted revision logging. They can be determined using the list command from other utilities (such as the ProcessAsset utility) or possibly by referring to existing log files:

<asset>	The moniker of the asset.
<type>	The moniker of the asset type.
<namespace>	The moniker namespace.
<attribute>	The moniker of the attribute.
<tagName>	The moniker of the tag.
<tagTypeName>	The moniker of the tag type.

## 9.4 Targeted Examples

```
[BASEDIR]/bin/ProcessRevisionHistoryExport -user admin
-export -logFile /var/opt/HMF/logs/myRevisions.log
-startTime 2009/07/08:09:10:11 -asset twilight system hmf
```

```
[BASEDIR]/bin/ProcessRevisionHistoryExport -user admin
-export -logFile /var/opt/HMF/logs/myRevisions.log
-startTime 2009/07/08:09:10:11 -assetType action hmf -assetType rule hmf
```

## 9.5 Notes

This targeted functionality is only available for the CMDB.

While this targets revisions of specified assets, asset attributes and tags, other entries may be exported if they happened within the same revision. For instance, if os asset twilight was created within the same revision (database transaction) as system asset twilight, both will be logged even if you indicate you only want to see system asset revisions. This is done because everything within a single revision is inherently linked to each other.

If the specified asset, asset attribute, or tag no longer exists, no entries will be exported.

# 10 Execute Operation

The Neuron Execute Operation utility can be used to execute operations that are defined in Neuron Management Suite (see Process Asset for adding/removing operations to/from Neuron Management Suite).

To execute an operation execute the following:

```
$ [BASEDIR]/bin/ExecuteOperation -user USER
    -asset ASSET TYPE NAMESPACE
    -operation OPERATION
    [-credentials CREDENTIALS]
    [-strategy STRATEGY]
    [-version VERSION]
    [VARIABLE=VALUE ...]
```

## 10.1 Supported Parameters

The following parameters are supported:

ASSET	The asset for which or on which to execute the operation.
TYPE	The type of asset specified above.
NAMESPACE	The namespace of the asset type specified above.
CREDENTIALS	The optional user name and encrypted password for operations that require additional user credentials. The SSH command requires this parameter if the asset does not contain a credential in the database.
STRATEGY	The strategy for selecting the job. This can be one of SSH or SCRIPT. If this parameter is provided the job with this strategy is selected. If it does not exist the operation cannot be executed.
VERSION	The version for selecting the job. If this parameter is provided the job with this version is selected. If it does not exist the operation cannot be executed.
VARIABLE	The name of a variable for the operation.
VALUE	The value of a variable for the operation.

**NOTE:** Only job variables of scope PUBLIC can be overwritten on the command-line in this way. PRIVATE and PROTECTED variables are for internal use by the job or service only. The scope is specified when a variable is added to a job.

## 10.2 Examples

```
[BASEDIR]/bin/ExecuteOperation -user admin
    -asset hostname system hmf -operation executeSSH
```

```
[BASEDIR]/bin/ExecuteOperation -user admin
  -asset hostname system hmf -operation executeSSH
  -credentials root LBzszxjlUSI= arg1=value1 arg2=value2
```

## 10.3 Command Services

This version of Neuron Management Suite supports the following command services (the service must be specified when adding a command using the Process Asset utility):

**SSH**                                      Execute an SSH command on the specified asset. This command requires the following operation arguments to be defined in the CMDB:

                                          command:    The name of the command to execute via SSH.  
                                          arguments:   The optional arguments for the SSH command.

**Script**                                    Execute a script on the Neuron Management Suite server. This command requires the following operation arguments to be defined in the CMDB:

                                          path:        The path of the command to execute. This can be a relative or absolute path. If the path is relative the command must be in [LOCALDIR]/bin.

                                          Any other arguments defined for the command are added as arguments to the command-line of the script.

## 10.4 How it Works

The Execute Operation sends a request to Neuron Management Suite in order to execute the operation. Neuron Management Suite first determines which operation to execute based on the supplied information. It then determines the job to execute. For this release only one job per operation may be specified. Neuron Management Suite then executes the job's commands in the following order: PRE\_RUN, RUN, POST\_RUN. If any of the job's commands fail execution stops and the result is returned to the caller. Upon successful execution of all commands the result is displayed.

# 11 Message Gateway

The Neuron Message Gateway can be used to inject messages into Neuron Management Suite. This is helpful for testing the correct setup of Neuron Management Suite.

To inject a single message into Neuron Management Suite execute the following commands:

```
% su -
# [BASEDIR]/bin/MessageGateway -user USER -topic TOPIC
    -type TYPE MESSAGE

# [BASEDIR]/bin/MessageGateway -type license -module MODULE VERSION DESCRIPTION
```

## 11.1 Supported Parameters

The following parameters are supported:

TOPIC	The topic on which to send the message. This can be one of GENERIC, EVENT, MODULE, or AMF. Use GENERIC to inject a message that will be logged by the Message Logger. Use EVENT to inject a message of type event (see TYPE). The topics MODULE and AMF are used by Neuron Management Suite for communicating with installed modules or the Neuron Management Portal.
TYPE	The type of message to send. This can be one of disc, test, event, or license. Message of type disc can be used to initiate discovery of assets (not supported in this release). The type test can be used in order to send a test message on the topic GENERIC to the Message Logger module. The type event should be used to inject an event on the topic EVENT. This will create a log event which will be displayed in the Neuron Event Viewer. Finally, using type license will create a trial-license for the module specified. This requires MODULE, VERSION, and DESCRIPTION be specified as well.
MESSAGE	The message to send. If type test is used then this message will be logged by the Message Logger. For type event this message will be part of the log event.

## 11.2 Injecting Multiple Messages

In order to inject multiple messages at once execute the following commands:

```
% su -
# [BASEDIR]/bin/MessageGateway -user USER -xml FILE
```

The FILE can be any XML file that adheres to the Neuron Management Suite schema hmf.xsd. This schema can be found in the META-INF directory of the Neuron Management Suite Common JAR file [BASEDIR]/lib/HMFCommon.jar.

The following is an example XML file containing one message for injecting a raw event into Neuron Management Suite:

```
<messages
  topic="EVENT"
  xmlns="http://www.halcyoninc.com/xml/ns/hmf"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.halcyoninc.com/xml/ns/hmf hmf.xsd">

  <rawEvent>
    <domain>default</domain>
    <vendor>Halcyon</vendor>
    <asset>fitzroy</asset>
    <eventType>FAULT</eventType>
    <eventMessage>Test Event</eventMessage>
    <eventState>OPEN</eventState>
    <eventSeverity>INFO</eventSeverity>
    <eventSource>rawEvent.xml</eventSource>
    <sourceProtocol>XML</sourceProtocol>
    <receivedTimestamp>2009-08-05 11:57:12.372842 GMT-0400</receivedTimestamp>
    <externalEventId>1</externalEventId>
  </rawEvent>
</messages>
```

## 11.3 Examples

```
MessageGateway -user admin -topic GENERIC
               -type test "Some test message"
```

```
MessageGateway -topic EVENT -type event "Test Event"
```

```
MessageGateway -xml rawEvent.xml
```

```
MessageGateway -type license -module MyModule 1.0 "Description of my module"
```

**NOTE:** No user and password are necessary in the following cases:

- Type is event or license.
- XML file contains only events.